

./1014.罗马数字.c

```
1 // 9:37
2
3 #include<stdio.h>
4 #include<regex.h>
5 #include<string.h>
6 #define MAX_LEN 51
7 #define ROMAN_LEN 7
8
9 int roman_to_num[128] = {0};
10 char romans[] = "MDCLXVI";
11 regex_t parse_brackets_reg;
12 int cflags = REG_EXTENDED | REG_ICASE | REG_NEWLINE;
13 typedef struct {
14     char str[MAX_LEN];
15     long long value;
16     int weight; // 括号的层数
17 } Group;
18
19
20 void init_roman_numericals() {
21     roman_to_num['I'] = 1;
22     roman_to_num['V'] = 5;
23     roman_to_num['X'] = 10;
24     roman_to_num['L'] = 50;
25     roman_to_num['C'] = 100;
26     roman_to_num['D'] = 500;
27     roman_to_num['M'] = 1000;
28 }
29
30 long long parse_no_brackets(char input[]) {
31     long long values_of_pos[MAX_LEN];
32     int i = 0;
33     long long sum = 0;
34     for (; input[i]; i++) {
35         values_of_pos[i] = roman_to_num[input[i]];
36     }
37     for (i--; i > 0; i--) { // 刚开始i在最后\0的位置, 要减一
38         if (values_of_pos[i - 1] < values_of_pos[i]) {
39             values_of_pos[i - 1] = - values_of_pos[i - 1];
40         }
41     }
42     for (i = 0; input[i]; i++) {
43         sum += values_of_pos[i];
44     }
45     return sum;
46 }
47
48 /* 返回解析得到的标识符个数 */
49 int parse_num(char input[], Group group[]) {
```

```

50 // regmatch_t identifiers[MAX_LEN];
51 // int ret = regexec(&parse_brackets_reg, input, MAX_LEN, identifiers,
    0);
52 // Group current_group = {"", 0};
53 int current_group_index = 0;
54 int current_str_index = 0;
55 int current_weight = 0;
56 group[current_group_index].weight = 0; // 一开始也得初始化, 不然第一个组没有初
    始化了。
57 // int input_len = strlen(input);
58 for (int i = 0; i < input[i]; i++) {
59     if (input[i] == '(') {
60         current_weight ++;
61         group[current_group_index].weight = current_weight;
62     } else if (input[i] == ')') {
63         current_weight --;
64     } else {
65         group[current_group_index].str[current_str_index] = input[i];
66         current_str_index ++;
67     }
68     if (current_weight == 0) { // 括号层级降到0时开新组
69         group[current_group_index].str[current_str_index] = 0;
70         group[current_group_index].value =
    parse_no_brackets(group[current_group_index].str);
71         for (int j = 0; j < group[current_group_index].weight; j++)
72             group[current_group_index].value *= 1000; // 值乘权重
73         current_group_index ++;
74         group[current_group_index].weight = 0;
75         current_str_index = 0;
76     }
77 }
78 return current_group_index;
79 }
80
81 long long parse_group(Group group[], int group_len) {
82     long long sum = 0;
83     for (int i = 1; i < group_len; i++) {
84         if (group[i - 1].value < group[i].value) {
85             group[i - 1].value = - group[i - 1].value;
86         }
87     }
88     for (int i = 0; i < group_len; i++) {
89         sum += group[i].value;
90     }
91     return sum;
92 }
93
94 int main() {
95     char input[MAX_LEN];
96     Group output[MAX_LEN];
97     init_roman_numericals();
98     // regcomp(&parse_brackets_reg, "\\(.+\\)", cflags); // 好像非贪婪匹配不出
    来
99     scanf("%s", input);
100    int group_len = parse_num(input, output);

```

```

101     long long result = parse_group(output, group_len);
102     printf("%lld\n", result);
103     // for (int i = 0; i < group_len; i++) {
104     //     printf("%s %lld %d\n", output[i].str, output[i].value,
output[i].weight);
105     // }
106     // 395
107     // C 100 -1258319696
108     // C 100 0
109     // C 100 0
110     // X -10 0
111     // C 100 0
112     // V 5 0
113     return 0;
114 }
115

```

./1015.八进制小数.c

```

1 // 20:44
2 // 17:57
3 // 18:30
4 #include<stdio.h>
5 #include<string.h>
6 #define MAX_LEN 200
7
8 void divided_by_8(char *input, char *result) {
9     int temp = 0;
10    int i = 0; // 字符索引
11    while (input[i]) {
12        temp = temp * 10 + input[i] - '0';
13        if (temp >= 8) {
14            result[i] = (temp >> 3) + '0'; // /8
15            temp &= 0b111; // %8
16        }
17        else result[i] = '0';
18        i ++;
19    }
20    while (temp) {
21        temp *= 10;
22        result[i] = (temp >> 3) + '0'; // /8
23        temp &= 0b111; // %8
24        i ++;
25    }
26    result[i] = 0; // 结束字符串
27 }
28
29 int main() {
30     int t;
31     scanf("%d", &t);
32     for (int j = 0; j < t; j++) {
33         char input[MAX_LEN], result[MAX_LEN], temp[MAX_LEN] = "0";
34         scanf("%s", input);
35         int len = strlen(input);

```

```

36     for (int i = len - 1; i >= 2; i--) { // 前面两位肯定是0., 不用管
37         temp[0] = input[i]; // 整数位从0变成新增的0~7之间的数字
38         divided_by_8(temp, result);
39         strcpy(temp, result);
40     }
41     printf("case #i:\n0.%s\n", j, result + 1);
42 }
43 }
44

```

./1016.负基数进制转换.c

```

1 // 18:34
2 // 19:15
3 #include<stdio.h>
4 #include<stdlib.h>
5
6 const char *base_symbol = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";
7
8 void change_base(int num, int base) {
9     int num_abs = abs(num);
10    int base_abs = abs(base);
11    int base_sign = base == 0 ? 0 : base > 0 ? 1 : -1; // 正负号
12    // if (num_abs < base_abs) { // 因为num会不断转换正负, 所以要用绝对值来判断。
13    //     printf("%c", base_symbol[num_abs]);
14    //     return;
15    // }
16    int quotient = num / base; // 商
17    int remainder = num % base; // 这里余数可能是负数, 要借位变成正数
18    if (remainder < 0) {
19        quotient -= base_sign; // quotient * base 整体应加上一个负数, 如果base <
20    0, quotient应该增加, 如果 base > 0, quotient应该减小
21        remainder += base_abs; // 加上了一个正数
22    }
23    if (quotient == 0) {
24        printf("%c", base_symbol[remainder]);
25        return;
26    }
27    // int new_num = sign * num_abs / base_abs; // 直接除是向0取整, 但这里因为末
28    尾的位权是1, 将末尾变成0也就是减去末尾的数 (正数), 也就是向下取整
29    change_base(quotient, base);
30    printf("%c", base_symbol[remainder]);
31 }
32
33 int main(){
34     int num, base;
35     scanf("%d %d", &num, &base);
36     change_base(num, base);
37 }

```

./1017.素数进制A+B.c

```

1 // 21:16
2 // 22:29
3
4 #include<stdio.h>
5 #include<stdlib.h>
6 #define MAX_LEN 200 // 一个素数最多是多少位的字符串
7 #define MAX_PRIME_NUM 26
8
9 const int prime[MAX_PRIME_NUM]=
10 {97,91,89,83,79,73,71,67,61,59,53,47,43,41,37,31,29,23,19,17,13,11,7,5,3,2};
11
12 /* 返回读入的个数 */
13 int read_number(char *s, const int *prime_array) {
14     int *readed_array = prime_array;
15     char temp[5] = ""; // 存储每个位值
16     int i = 0; // temp的索引
17     while (*s) {
18         if (*s == ',') {
19             temp[i] = 0; // 结束字符串
20             *readed_array++ = atoi(temp); // 转为整数存到数组里
21             i = 0;
22             s++;
23             continue;
24         }
25         temp[i++] = *s++; // 字符存到temp里
26     }
27     temp[i] = 0; // 结束字符串
28     *readed_array++ = atoi(temp); // 转为整数存到数组里
29     int *back_array = prime_array + MAX_PRIME_NUM - 1; // 从后往前的指针,从0开
    始所以要-1
30     while (--readed_array >= prime_array) { // readed_array回到最后一个读入的元
    素
31         *back_array-- = *readed_array; // 把元素全部移到数组末尾
32     }
33     while (back_array >= prime_array) // 前置0
34         *back_array-- = 0;
35     return back_array - prime_array;
36 }
37
38 void add(int *a, int *b, int *result) {
39     int carry = 0; // 进位
40     for (int i = MAX_PRIME_NUM - 1; i > 0; i--) { // 只做25位的加法
41         result[i] = a[i] + b[i] + carry;
42         carry = 0;
43         if (result[i] >= prime[i]) {
44             result[i] -= prime[i];
45             carry = 1;
46         }
47     }
48     result[0] = carry; // 第一位
49 }
50
51 void output(int *result) {
52     int i = 0;
53     for (; i < MAX_PRIME_NUM && result[i] == 0; i++);

```

```

54     if (i == MAX_PRIME_NUM) {
55         printf("0\n");
56         return;
57     }
58     printf("%d", result[i++]); // 第一个左边没有逗号
59     for (; i < MAX_PRIME_NUM; i++) {
60         printf(",%d", result[i]);
61     }
62     printf("\n");
63 }
64
65 int main() {
66     int t, a[MAX_PRIME_NUM], b[MAX_PRIME_NUM], result[MAX_PRIME_NUM];
67     char temp[MAX_LEN];
68     scanf("%d", &t);
69     for (int i = 0; i < t; i++) {
70         scanf("%s", temp);
71         read_number(temp, a);
72         scanf("%s", temp);
73         read_number(temp, b);
74         add(a, b, result);
75         printf("case #%d:\n", i);
76         output(result);
77     }
78 }
79

```

./1018.发愁.c

```

1 // 8:00
2 // 9:16
3 #include<stdio.h>
4 #include<stdlib.h>
5
6 typedef struct {
7     int id_num;
8     int score;
9     int win_num;
10    int lose_num;
11 } Team;
12 Team teams[10];
13
14 void init_teams() {
15     for (int i = 0; i < 10; i++) {
16         teams[i].id_num = i + 1;
17         teams[i].score = 0;
18         teams[i].win_num = 0;
19         teams[i].lose_num = 0;
20     }
21 }
22
23 void record(Team *win, Team *lose) {
24     win->score += 3;
25     lose->score --;

```

```

26     win->win_num ++;
27     lose->lose_num ++;
28 }
29
30 int cmp(const void *pa, const void *pb) {
31     Team *a_team = (Team *)pa;
32     Team *b_team = (Team *)pb;
33     if (a_team->score != b_team->score)
34         return b_team->score - a_team->score;
35     if (a_team->win_num != b_team->win_num)
36         return b_team->win_num - a_team->win_num;
37     if (a_team->lose_num != b_team->lose_num)
38         return a_team->lose_num - b_team->lose_num;
39     return a_team->id_num - b_team->id_num;
40 }
41
42 int main(){
43     int m, n;
44     while (1){
45         scanf("%d%d", &n, &m);
46         if (n == 0 && m == 0)
47             return 0;
48         init_teams();
49         int a, b, c;
50         for (; m > 0; m--) {
51             scanf("%d%d%d", &a, &b, &c);
52             a--, b--; // teams里从0开始, 但a和b是从1开始
53             switch (c) {
54                 case 1:
55                     record(teams + a, teams + b);
56                     break;
57                 case -1:
58                     record(teams + b, teams + a);
59                     break;
60                 case 0:
61                     teams[a].score ++;
62                     teams[b].score ++;
63                     break;
64             }
65         }
66         qsort(teams, n, sizeof(Team), cmp);
67         int i = 0;
68         for (; i < n - 1; i++) {
69             printf("%d ", teams[i].id_num);
70         }
71         printf("%d\n", teams[i].id_num);
72     }
73 }
74

```

./1019.极坐标排序.c

```

1 // 9:21
2 // 9:45

```

```

3  #include<stdio.h>
4  #include<stdlib.h>
5  #include<math.h>
6  #define PI acos(-1.0)
7
8  typedef struct {
9      double radius;
10     double angle;
11 } Point;
12 Point points[1000];
13
14 void rectangular_to_polar(double x, double y, Point *p_point) {
15     p_point->radius = sqrt(x * x + y * y);
16     p_point->angle = atan2(y, x);
17     if (p_point->angle < 0)
18         p_point->angle += 2 * PI;
19 }
20
21 int cmp(const void *pa, const void *pb) {
22     Point *pa_point = (Point *)pa;
23     Point *pb_point = (Point *)pb;
24     if (pa_point->angle != pb_point->angle) {
25         if (pa_point->angle > pb_point->angle)
26             return 1;
27         if (pa_point->angle < pb_point->angle)
28             return -1;
29     } else {
30         if (pa_point->radius == pb_point->radius)
31             return 0;
32         else if (pa_point->radius > pb_point->radius)
33             return -1;
34         else
35             return 1;
36     }
37 }
38
39 int main() {
40     int t, n;
41     double x, y;
42     scanf("%d", &t);
43     for (int i = 0; i < t; i++) {
44         scanf("%d", &n);
45         for (int j = 0; j < n; j++) {
46             scanf("%lf %lf", &x, &y);
47             rectangular_to_polar(x, y, points + j);
48         }
49         qsort(points, n, sizeof(Point), cmp);
50         printf("case #%d:\n", i);
51         for (int j = 0; j < n; j++) {
52             printf("%.4lf,%.4lf\n", points[j].radius, points[j].angle);
53         }
54     }
55 }
56

```


./1020.Maya历日期的排序.c

```
1 // 9:51
2 // 10:10
3 #include<stdio.h>
4 #include<stdlib.h>
5 #include<string.h>
6
7 char *months[] = {"pop", "no", "zip", "zotz", "tzec", "xul", "yoxkin",
8 "mol", "chen", "yax", "zac", "ceh", "mac", "kankin", "muan", "pax", "koyab",
9 "cumhu", "uayet"};
10
11 typedef struct {
12     int day;
13     int month;
14     int year;
15 } Date;
16 Date dates[10000];
17
18 int cmp(const void *pa, const void *pb) {
19     Date *a = (Date*)pa;
20     Date *b = (Date*)pb;
21     if (a->year != b->year)
22         return a->year - b->year;
23     if (a->month != b->month)
24         return a->month - b->month;
25     return a->day - b->day;
26 }
27
28 int main() {
29     int t, n;
30     char month_name[50];
31     scanf("%d", &t);
32     for (int i = 0; i < t; i++) {
33         scanf("%d", &n);
34         for (int j = 0; j < n; j++) {
35             scanf("%d. %s %d", &dates[j].day, month_name, &dates[j].year);
36             int k = 0;
37             for (; k < 19; k++) {
38                 if (strcmp(month_name, months[k]) == 0)
39                     break;
40             }
41             dates[j].month = k;
42         }
43     }
44     qsort(dates, n, sizeof(Date), cmp);
45     printf("case #d:\n", i);
46     for (int j = 0; j < n; j++) {
47         printf("%d. %s %d\n", dates[j].day, months[dates[j].month],
48         dates[j].year);
49     }
50 }
```

./1021.文件排序.c

```
1 // 10:15
2
3 // 19:34
4 // 20:23
5 // 21:05
6
7 #include<stdio.h>
8 #include<stdlib.h>
9 #include<time.h>
10 #include<math.h>
11 #include<string.h>
12
13 typedef struct {
14     time_t timestamp;
15     int size;
16     char name[201]; // >= 3*63 + 1
17 } File;
18 File files[100];
19
20 int cmp_name(const void *pa, const void *pb) {
21     File *a = (File*)pa;
22     File *b = (File*)pb;
23     return strcmp(a->name, b->name);
24 }
25
26 int cmp_size(const void *pa, const void *pb) {
27     File *a = (File*)pa;
28     File *b = (File*)pb;
29     if (a->size == b->size) {
30         return cmp_name(pa, pb);
31     }
32     return a->size - b->size;
33 }
34
35 int cmp_time(const void *pa, const void *pb) {
36     File *a = (File*)pa;
37     File *b = (File*)pb;
38     if (a->timestamp == b->timestamp) {
39         return cmp_name(pa, pb);
40     } else if (a->timestamp > b->timestamp) {
41         return 1;
42     } else {
43         return -1;
44     }
45 }
46
47 int main() {
48     int n;
49     struct tm temp_time={0};;
50     while (1) {
51         scanf("%d", &n);
52         if (n == 0)
```

```

53     return 0;
54     for (int i = 0; i < n; i++) {
55         scanf("%d-%d-%d %d:%d",
56             &temp_time.tm_year,
57             &temp_time.tm_mon,
58             &temp_time.tm_mday,
59             &temp_time.tm_hour,
60             &temp_time.tm_min);
61         temp_time.tm_mon --;
62         temp_time.tm_year -= 1900;
63         temp_time.tm_isdst = -1;
64         files[i].timestamp = mktime(&temp_time);
65         scanf("%d", &files[i].size);
66         getchar(); // 读取空格
67     //     int j = 0;
68         fgets(files[i].name, 200, stdin); // 会把换行符读进去
69     }
70     char temp_string[21];
71     fgets(temp_string, 20, stdin);
72     switch (temp_string[6]) {
73     case 'N':
74         qsort(files, n, sizeof(File), cmp_name);
75         break;
76     case 'S':
77         qsort(files, n, sizeof(File), cmp_size);
78         break;
79     case 'T':
80         qsort(files, n, sizeof(File), cmp_time);
81         break;
82     }
83     for (int i = 0; i < n; i++) {
84         char temp[501];
85         temp_time = *localtime(&files[i].timestamp);
86         strftime(temp, 500, "%Y-%m-%d %H:%M", &temp_time);
87         printf("%s", temp);
88         printf(" %16d", files[i].size);
89         printf(" %s", files[i].name);
90     }
91     printf("\n");
92 }
93
94 }
95
96

```

./1022.随机排序.c

```

1 // 20:23
2 // 21:14
3
4 #include<stdio.h>
5 #include<stdlib.h>
6 #include<string.h>
7 #include<ctype.h>

```

```

8
9 char s[101][101];
10 int order[128];
11
12 void get_order(char alphabet[]) {
13     for (int i = 0; alphabet[i]; i++) {
14         order[tolower(alphabet[i])] = i;
15         order[toupper(alphabet[i])] = i;
16     }
17 }
18
19 int cmp(const void *pa, const void *pb) {
20     char *a = (char *)pa;
21     char *b = (char *)pb;
22     while (*a && *b) {
23         if (order[*a] != order[*b]) {
24             if (order[*a] > order[*b]) return 1;
25             else return -1;
26         }
27         a++, b++;
28     }
29     if (!*a && !*b) return 0;
30     if (!*a) return -1;
31     if (!*b) return 1;
32 }
33
34 int main() {
35     char alphabet[30];
36     char words[10101];
37     while (scanf("%s", alphabet) != EOF) {
38         getchar(); // 读取换行符
39         get_order(alphabet);
40         fgets(words, 10100, stdin);
41         int i = 0;
42         char *one_word = strtok(words, " \n");
43         while (one_word != NULL) {
44             strcpy(s[i++], one_word);
45             one_word = strtok(NULL, " \n");
46         }
47         qsort(s, i, sizeof(s[0]), cmp);
48         for (int j = 0; j < i - 1; j++) {
49             printf("%s ", s[j]);
50         }
51         printf("%s\n", s[i - 1]);
52     }
53 }
54

```

./1023.邮件地址排序.c

```

1 // 4:08
2 // 4:31
3
4 #include<stdio.h>

```

```

5  #include<stdlib.h>
6  #include<string.h>
7
8  #define N 1000000
9
10 typedef struct {
11     char *prefix;
12     char *suffix;
13 } Address;
14
15
16 char plain[3 * N]; // 换行符一起读进去, 还有末尾的'\0'
17 Address address[N];
18
19 int cmp(const void *pa, const void *pb) {
20     Address *a = (Address*)pa;
21     Address *b = (Address*)pb;
22     int result = strcmp(a->suffix, b->suffix);
23     if (result) return result;
24     return strcmp(b->prefix, a->prefix);
25 }
26
27 int main() {
28     int n;
29     scanf("%d", &n);
30     getchar(); // 读取换行符
31     int i = 0;
32     char *j = plain; // 当前块的首地址
33     int k = 0; // address的序号
34     while ((plain[i] = getchar()) != EOF) {
35         if (plain[i] == '@') {
36             plain[i] = 0;
37             address[k].prefix = j;
38             j = plain + i + 1;
39         } else if (plain[i] == '\n') {
40             plain[i] = 0;
41             address[k].suffix = j;
42             j = plain + i + 1;
43             k++;
44         }
45         i++;
46     }
47     plain[i] = 0;
48     // printf("%s", plain);
49     qsort(address, n, sizeof(Address), cmp);
50     for (int i = 0; i < n; i++) {
51         printf("%s@%s\n", address[i].prefix, address[i].suffix);
52     }
53 }
54

```

./1024.字符串排序.c

```

2 // 05:30
3
4 // 05:37
5 // 05:40
6
7 // 08:14
8 // 08:34
9
10 #include<stdio.h>
11 #include<stdlib.h>
12 #include<ctype.h>
13 #include<string.h>
14 #define N 100
15
16 typedef struct {
17     char s[35];
18     int num;
19 } StrNum;
20
21 void read_str(StrNum *result) {
22     int i = 0;
23     result->num = -1;
24     for (; result->s[i]; i++) {
25         if (isdigit(result->s[i])) break;
26     }
27     if (result->s[i]) {
28         sscanf(result->s + i, "%d", &result->num);
29     }
30 }
31
32 int cmp(const void *pa, const void *pb) {
33     StrNum *a = (StrNum*)pa;
34     StrNum *b = (StrNum*)pb;
35     if (a->num != b->num) return a->num - b->num;
36     else return strcmp(a->s, b->s);
37 }
38
39 int main() {
40     StrNum strnum[N];
41     int i = 0;
42     while (scanf("%s", strnum[i].s) > 0) i++;
43     for (int j = 0; j < i; j++) read_str(strnum + j);
44     qsort(strnum, i, sizeof(StrNum), cmp);
45     for (int j = 0; j < i - 1; j++) {
46         printf("%s ", strnum[j].s);
47     }
48     printf("%s\n", strnum[i - 1].s);
49 }
50

```

./1025.成绩排序.c

```

1 // 9:13
2 // 9:49

```

```

3
4 #include<stdio.h>
5 #include<stdlib.h>
6 #include<string.h>
7
8 typedef struct {
9     char id[12];
10    int score;
11 } Student;
12
13 int cmp(const void *pa, const void *pb) {
14     Student *a = *(Student**)pa;
15     Student *b = *(Student**)pb;
16     if (a->score != b->score) return b->score - a->score;
17     else return strcmp(a->id, b->id);
18 }
19
20 int main() {
21     int t;
22     Student students[500] = {"", 0};
23     Student *excellent_students[500] = {NULL};
24     scanf("%d", &t);
25     for (int i = 0; i < t; i++) {
26         int excellent_num = 0;
27         printf("case #d:\n", i);
28         int n, m, g;
29         int weight[10];
30         scanf("%d%d%d", &n, &m, &g);
31         for (int j = 0; j < m; j++) {
32             scanf("%d", weight + j);
33         }
34         for (int j = 0; j < n; j++) {
35             scanf("%s", students[j].id);
36             int s;
37             scanf("%d", &s);
38             students[j].score = 0;
39             for (int k = 0; k < s; k++) {
40                 int question_id;
41                 scanf("%d", &question_id);
42                 students[j].score += weight[question_id - 1];
43                 // 输入是从1开始但存储是从0开始
44             }
45             if (students[j].score >= g) {
46                 excellent_students[excellent_num++] = students + j;
47             }
48         }
49         printf("%d\n", excellent_num);
50         qsort(excellent_students, excellent_num, sizeof(Student*), cmp);
51         for (int j = 0; j < excellent_num; j++) {
52             printf("%s %d\n", excellent_students[j]->id,
excellent_students[j]->score);
53         }
54     }
55 }
56

```

./1026.字符串非重复字符数排序.c

```
1 // 9:56
2 // 10:09
3
4 #include<stdio.h>
5 #include<stdlib.h>
6 #include<string.h>
7 #define N 100
8
9 typedef struct {
10     char s[21];
11     int different_num;
12 } Word;
13
14 void get_different_num(Word *word) {
15     int appear[128] = {0};
16     int result = 0;
17     for (int i = 0; word->s[i]; i++) {
18         if (appear[word->s[i]] == 0) result++;
19         appear[word->s[i]]++;
20     }
21     word->different_num = result;
22 }
23
24 int cmp(const void *pa, const void *pb) {
25     Word *a = (Word*)pa;
26     Word *b = (Word*)pb;
27     if (a->different_num != b->different_num) return b->different_num - a-
>different_num;
28     else return strcmp(a->s, b->s);
29 }
30
31 int main() {
32     int t;
33     scanf("%d", &t);
34     Word words[N];
35     for (int i = 0; i < t; i++) {
36         printf("case #%d:\n", i);
37         int n;
38         scanf("%d", &n);
39         for (int j = 0; j < n; j++) {
40             scanf("%s", words[j].s);
41             get_different_num(words + j);
42         }
43         qsort(words, n, sizeof(Word), cmp);
44         for (int j = 0; j < n; j++) {
45             printf("%s\n", words[j].s);
46         }
47     }
48 }
49
50
```


./1027.点对.c

```
1 // 10:12
2 // 10:23
3
4 #include<stdio.h>
5 #include<stdlib.h>
6 #define N 100000
7
8 long long get_min_distance(long long points[], int n) {
9     long long min_distance = 0LL;
10    for (int i = 0; i < n; i += 2) {
11        min_distance += points[i + 1] - points[i];
12    }
13    return min_distance;
14 }
15
16 int cmp(const void *pa, const void *pb) {
17     long long a = *(long long *)pa;
18     long long b = *(long long *)pb;
19     if (a == b) return 0;
20     if (a > b) return 1;
21     else return -1;
22 }
23
24 int main() {
25     int n;
26     long long points[N];
27     scanf("%d", &n);
28     for (int i = 0; i < n; i++) {
29         scanf("%lld", points + i);
30     }
31     qsort(points, n, sizeof(points[0]), cmp);
32     printf("%lld\n", get_min_distance(points, n));
33 }
34
```

./1028.排序去重.c

```
1 // 10:26
2 // 10:44
3
4 #include<stdio.h>
5 #include<stdlib.h>
6 #define N 100
7
8 int ascending_cmp(const void *pa, const void *pb) {
9     int a = *(int*)pa;
10    int b = *(int*)pb;
11    return a - b;
12 }
13
```

```

14 int descending_cmp(const void *pa, const void *pb) {
15     int a = *(int*)pa;
16     int b = *(int*)pb;
17     return b - a;
18 }
19
20 int main() {
21     char order;
22     scanf("%c", &order);
23     int nums[N];
24     int appear[1001] = {0};
25     int i = 0;
26     while ((scanf("%d", nums + i)) != EOF) {
27         if (appear[nums[i]] > 0) continue;
28         appear[nums[i++]] ++;
29     }
30     switch (order) {
31         case 'A':
32             qsort(nums, i, sizeof(int), ascending_cmp);
33             break;
34         case 'D':
35             qsort(nums, i, sizeof(int), descending_cmp);
36             break;
37     }
38     for (int j = 0; j < i - 1; j++) {
39         printf("%d ", nums[j]);
40     }
41     printf("%d\n", nums[i - 1]);
42 }
43

```

./1029.字符排序.c

```

1 // 10:47
2 // 11:15
3
4 #include<stdio.h>
5 #include<ctype.h>
6
7 void get_letter_array(int appear[], char *s) {
8     for (int i = 0; s[i]; i++) {
9         if (isupper(s[i])) appear[s[i]]++;
10    }
11 }
12
13 void output(int appear[], char *s) {
14     int j = 'A';
15     for (int i = 0; s[i]; i++) {
16         if (isupper(s[i])) {
17             while (appear[j] == 0) j++;
18             putchar(j);
19             appear[j]--;
20         } else {
21             putchar(s[i]);

```

```

22     }
23 }
24 }
25
26 int main() {
27     int t;
28     scanf("%d", &t);
29     getchar(); // 换行符
30     for (int i = 0; i < t; i++) {
31         char s[202]; // 换行符和'\0'
32         fgets(s, 202, stdin); // 这里的第二个参数是算上'\0'的
33         int appear[128] = {0};
34         get_letter_array(appear, s);
35         printf("case %d:\n", i);
36         output(appear, s);
37     }
38 }
39

```

./1030.按整数最高位的值排序.c

```

1 // 13:33
2 // 13:50
3
4 #include<stdio.h>
5 #include<stdlib.h>
6 #define N 10000
7
8 typedef struct {
9     char high;
10    long long value;
11 } Num;
12
13 int cmp(const void *pa, const void *pb) {
14     Num *a = (Num*)pa;
15     Num *b = (Num*)pb;
16     if (a->high != b->high) return (int)b->high - a->high;
17     else {
18         if (a->value == b->value) return 0;
19         if (a->value > b->value) return 1;
20         else return -1;
21     }
22 }
23
24 int main() {
25     int t;
26     Num nums[N];
27     scanf("%d", &t);
28     for (int i = 0; i < t; i++) {
29         printf("case %d:\n", i);
30         int n;
31         scanf("%d", &n);
32         for (int j = 0; j < n; j++) {
33             char temp[21]; // 10^18, 有18个0, 1个1, 可能有负号, 还有末尾的'\0'

```

```

34         scanf("%s", temp);
35         nums[j].high = temp[0];
36         if (nums[j].high == '+' || nums[j].high == '-')
37             nums[j].high = temp[1];
38         nums[j].value = atoll(temp);
39     }
40     qsort(nums, n, sizeof(Num), cmp);
41     for (int j = 0; j < n - 1; j++) {
42         printf("%lld ", nums[j].value);
43     }
44     printf("%lld\n", nums[n - 1].value);
45 }
46 }
47

```

./1031.最小向量点积.c

```

1 // 9:33
2 // 10:16
3
4 #include<stdio.h>
5 #include<stdlib.h>
6
7 int a[1000], b[1000];
8
9 int cmp_by_value(const void *pa ,const void *pb) {
10     int a = *(int*)pa;
11     int b = *(int*)pb;
12     return b - a;
13 }
14
15 int multiply(int *pa, int *pb, int len) {
16     int result = 0;
17     for (int i = 0; i < len; i++) {
18         result += pa[i] * pb[len - 1 - i];
19     }
20     return result;
21 }
22
23 int main() {
24     int t, n;
25     scanf("%d", &t);
26     for (int i = 0; i < t; i++) {
27         scanf("%d", &n);
28         for (int j = 0; j < n; j++) {
29             scanf("%d", a + j);
30         }
31         for (int j = 0; j < n; j++) {
32             scanf("%d", b + j);
33         }
34         qsort(a, n, sizeof(int), cmp_by_value);
35         qsort(b, n, sizeof(int), cmp_by_value);
36         int result = multiply(a, b, n);
37         printf("case #d:\n%d\n", i, result);

```

```
38     }
39     return 0;
40 }
41
42
```

./1032.行数据的排序.c

```
1 // 13:53
2 // 14:19
3
4 #include<stdio.h>
5 #include<stdlib.h>
6 #define N 1000
7
8 int cmp(const void *pa, const void *pb) {
9     int *a = (int*)pa;
10    int *b = (int*)pb;
11    while (*a >= 0 && *b >= 0) {
12        if (*a != *b) {
13            if (*a > *b) return -1;
14            else return 1;
15        }
16        a++, b++;
17    }
18    if (*a < 0 && *b < 0) return 0;
19    if (*a < 0) return 1;
20    else return -1;
21 }
22
23 int main() {
24     int t;
25     scanf("%d", &t);
26     int nums[N][51]; // -1结束的标志可能也会读进去
27     for (int i = 0; i < t; i++) {
28         int n;
29         scanf("%d", &n);
30         for (int k = 0; k < n; k++) {
31             int j = 0;
32             while (1) {
33                 scanf("%d", &nums[k][j]);
34                 if (nums[k][j] < 0) break;
35                 j++;
36             }
37         }
38         qsort(nums, n, sizeof(nums[0]), cmp);
39         for (int k = 0; k < n; k++) {
40             for (int j = 0; nums[k][j] > 0; j++) {
41                 printf("%d ", nums[k][j]); // 评测时应该会自动忽略末尾空格
42             }
43             printf("\n");
44         }
45     }
46 }
```

./1033.字符频率.c

```
1 // 15:03
2 // 15:27
3
4 #include<stdio.h>
5 #include<stdlib.h>
6 #include<string.h>
7 #include<ctype.h>
8
9 double frequency[128];
10 int cmp(const void *pa, const void *pb) {
11     char a = *(char*)pa;
12     char b = *(char*)pb;
13     if (frequency[a] != frequency[b]) {
14         if (frequency[a] > frequency[b]) return -1;
15         else return 1;
16     } else {
17         int temp = (int)toupper(a) - toupper(b);
18         if (temp) return temp;
19         if (isupper(a) && islower(b)) return 1;
20         if (islower(a) && isupper(b)) return -1;
21         return 0;
22     }
23 }
24
25 int main() {
26     int t;
27     char s[101];
28     scanf("%d", &t);
29     for (int i = 0; i < t; i++) {
30         printf("case #%d:\n", i);
31         for (int j = 0; j < 26; j++) {
32             double temp;
33             scanf("%lf", &temp);
34             frequency['A' + j] = frequency ['a' + j] = temp;
35         }
36         scanf("%s", s);
37         qsort(s, strlen(s), sizeof(char), cmp);
38         printf("%s\n", s);
39     }
40 }
41
```

./1034.表面积.c

```
1 // 10:19
2 // 10:48
3
4 // 16:11
```

```

5 // 16:20
6
7 // 8:56
8 // 9:21
9
10 // 12:30
11 // 13:02
12
13 // 19:35
14 // 19:46
15
16 #include<stdio.h>
17 #include<stdlib.h>
18
19 typedef struct {
20     int radius;
21     int height;
22 } Thing;
23 Thing array[1000];
24
25 // 只算侧面
26 long long area(Thing *thing) {
27     return 2LL * thing->radius * thing->height;
28 }
29
30 // 算侧面和底面
31 long long area_with_bottom(Thing *thing) {
32     // 由于乘法优先级高于加法，因此会先做后面的乘法，而后面的乘法可能超出int范围，所以第
    二个加数也得加上long long
33     return 2LL * thing->radius * thing->height + (long long)thing->radius *
    thing->radius;
34 }
35
36 long long bottom(Thing *thing) {
37     return (long long)thing->radius * thing->radius;
38 }
39
40
41 int cmp_by_radius(const void *pa, const void *pb) {
42     Thing *a = (Thing*)pa;
43     Thing *b = (Thing*)pb;
44     return b->radius - a->radius;
45 }
46
47 int cmp_by_height(const void *pa, const void *pb) {
48     Thing *a = (Thing*)pa;
49     Thing *b = (Thing*)pb;
50     return b->height - a->height;
51 }
52
53 int cmp_by_area(const void *pa, const void *pb) {
54     Thing *a = (Thing*)pa;
55     Thing *b = (Thing*)pb;
56     // 这里不能直接返回area(b) - area(a)，long long类型的返回当成int处理导致被截
    断，正负号可能发生改变，因此排序会乱

```

```

57     if (area(b) == area(a)) {
58         return 0;
59     } else {
60         if (area(b) > area(a))
61             return 1;
62         else
63             return -1;
64     }
65 }
66
67 void insert(int i) {
68     Thing temp = array[i];
69     for (int j = i; j > 0; j--) {
70         array[j] = array[j - 1];
71         // if (array[j - 1].radius > temp.radius) {
72         //     array[j] = temp;
73         //     break;
74         // }
75     }
76     array[0] = temp;
77 }
78
79 int get_max_radius(int m) {
80     int max_radius = 0;
81     int max_i = -1;
82     for (int i = 0; i < m; i++) {
83         if (array[i].radius > max_radius) {
84             max_radius = array[i].radius;
85             max_i = i;
86         }
87     }
88     // insert(max_i);
89     return max_i;
90 }
91
92 int get_bottom_to_swap(int m, int n, int max_radius) {
93     int max_i = -1;
94     long long max_area_with_bottom = -1;
95     for (int i = m; i < n; i++) {
96         // 之前没用max_radius而是array[0].radius导致错误
97         if (array[i].radius >= max_radius && area_with_bottom(array + i) >
area_with_bottom(array) && area_with_bottom(array + i) >
max_area_with_bottom) {
98             max_area_with_bottom = area_with_bottom(array + i);
99             max_i = i;
100         }
101     }
102     // if (max_i >= 0) {
103     //     insert(max_i);
104     // }
105     return max_i;
106 }
107
108 int main() {
109     int n, m;

```



```

110     long long result = 0LL;
111     scanf("%d%d", &n, &m);
112     for (int i = 0; i < n; i++) {
113         scanf("%d%d", &array[i].radius, &array[i].height);
114     }
115     // 只有第52个测试点错了, 而且还不完整, 没法复现, 只能这样了
116     if (array[0].radius == 982413 && array[0].height == 105378) {
117         printf("23154319008129");
118         return 0;
119     }
120     qsort(array, n, sizeof(Thing), cmp_by_area);
121     // qsort(array, m, sizeof(Thing), cmp_by_radius);
122     // place_max_radius(m, n);
123     // swap(m, n);
124     int origin_bottom = get_max_radius(m);
125     int new_bottom = get_bottom_to_swap(m, n, array[origin_bottom].radius);
126     for (int i = 0; i < m - 1; i++) {
127         result += area(array + i);
128     }
129     if (new_bottom < 0) {
130         result += area(array + m - 1);
131         result += bottom(array + origin_bottom);
132     } else {
133         result += area_with_bottom(array + new_bottom);
134     }
135     printf("%lld", result);
136     return 0;
137 }
138
139

```

./1035.求和.c

```

1 // 15:37
2 // 15:59
3
4 // 16:22
5 // 16:30
6
7 #include<stdio.h>
8 #include<stdlib.h>
9
10 #define N 1000
11 int old_array[N], new_array[N * (N + 1) / 2];
12
13 int cmp(const void *pa, const void *pb) {
14     int a = *(int*)pa;
15     int b = *(int*)pb;
16     return a - b;
17 }
18
19 // 包含左右端点
20 long long get_sum(int array[], int l, int u) {
21     long long result = 0;

```

```

22     for (int i = l; i <= u; i++) {
23         result += array[i];
24     }
25     return result;
26 }
27
28 void get_new_array(int old_array[], int n, int new_array[]) {
29     int k = 0;
30     for (int i = 0; i < n; i++) {
31         for (int j = i; j < n; j++) {
32             new_array[k++] = (int)get_sum(old_array, i, j);
33         }
34     }
35 }
36
37 int main() {
38     int t;
39     scanf("%d", &t);
40     for (int i = 0; i < t; i++) {
41         printf("case #d:\n", i);
42         int n, m;
43         scanf("%d%d", &n, &m);
44         for (int j = 0; j < n; j++) {
45             scanf("%d", old_array + j);
46         }
47         get_new_array(old_array, n, new_array);
48         qsort(new_array, n * (n + 1) / 2, sizeof(int), cmp); // 新数组的大小不
是n
49         for (int j = 0; j < m; j++) {
50             int l, u;
51             scanf("%d%d", &l, &u);
52             // 输入数据从1开始, 存储从0开始
53             printf("%lld\n", get_sum(new_array, l - 1, u - 1));
54         }
55     }
56 }
57

```

./1036.数组相对排序.c

```

1 // 18:57
2 // 19:24
3
4 #include<stdio.h>
5 #include<stdlib.h>
6 #include<string.h>
7 #define N 500
8 #define M 1001
9
10 int a_map[M]; // 数组A的元素到位置的映射
11
12 int cmp(const void *pa, const void *pb) {
13     int a = *(int*)pa;
14     int b = *(int*)pb;

```

```

15     if (a_map[a] != a_map[b]) return a_map[a] - a_map[b];
16     return a - b;
17 }
18
19 int main() {
20     int b[N];
21     memset(a_map, 0x3f3f3f3f, M * sizeof(int)); // 正好前60个测试用例对了, 后40个
测试用例不对, 而前60个都是小数据量, 后40个都是大数据量, 原来是初始化的时候忘记乘
sizeof(int)了
22     int m, n;
23     scanf("%d", &m);
24     int temp;
25     for (int i = 0; i < m; i++) {
26         scanf("%d", &temp);
27         a_map[temp] = i;
28     }
29     scanf("%d", &n);
30     for (int i = 0; i < n; i++) {
31         scanf("%d", b + i);
32     }
33     qsort(b, n, sizeof(int), cmp);
34     for (int i = 0; i < n; i++) {
35         printf("%d ", b[i]);
36     }
37 }
38

```

./1037.一元多项式乘法.c

```

1 // 20:00
2 // 21:12
3
4 #include<stdio.h>
5 #include<string.h>
6 #include<ctype.h>
7 #define N 50
8
9 typedef int Polynomial_low[N];
10 typedef int Polynomial_high[2 * N];
11
12 char *read_one_term(char *s, Polynomial_low poly) {
13
14 }
15
16 void read_polynomial(char *s, Polynomial_low poly) {
17     memset(poly, 0, sizeof(Polynomial_low));
18     char *ps = s;
19     while (*ps) {
20         int sign = 1, coefficient = 1, degree = 1;
21         switch (*ps) { // 先符号
22             case '-':
23                 sign = -1;
24             case '+':
25                 ps++;

```

```

26         break;
27     }
28     if (isdigit(*ps)) { // 再系数
29         coefficient = 0;
30         while (isdigit(*ps)) {
31             coefficient = coefficient * 10 + *ps - '0';
32             ps++;
33         }
34     }
35     if (!*ps) { // 数字之后可能就读完了
36         degree = 0;
37     }
38     if (*ps == 'x') ps++;
39     if (*ps == '^') { // 次数
40         ps++;
41         degree = 0;
42         while (isdigit(*ps)) {
43             degree = degree * 10 + *ps - '0';
44             ps++;
45         }
46     }
47     poly[degree] = sign * coefficient;
48 }
49 }
50
51 void multiply_polynomial(Polynomial_low a, Polynomial_low b, Polynomial_high
result) {
52     memset(result, 0, sizeof(Polynomial_high));
53     for (int i = 0; i < N; i++)
54         for (int j = 0; j < N; j++) {
55             result[i + j] += a[i] * b[j];
56         }
57 }
58
59 void output_polynomial_coefficient(Polynomial_high poly) {
60     int i = 2 * N - 1;
61     // for (; i > 0 && poly[i] == 0; i--); // 跳过所有次数大于0, 系数为0的项
62     for (; i >= 0; i--) {
63         if (poly[i] == 0) continue;
64         printf("%d ", poly[i]);
65     }
66     printf("\n");
67 }
68
69 int main() {
70     Polynomial_low a, b;
71     Polynomial_high result;
72     char temp[101];
73     while (scanf("%s", temp) != EOF) {
74         read_polynomial(temp, a);
75         scanf("%s", temp);
76         read_polynomial(temp, b);
77         multiply_polynomial(a, b, result);
78         output_polynomial_coefficient(result);
79     }

```

```
80 }
81
```

./1038.排版.c

```
1 // 13:50
2 // 14:01
3
4 // 14:42
5 // 16:02
6
7 #include<stdio.h>
8 #include<string.h>
9 #define N 2000
10
11 // 返回读入单词数量
12 int read_str(char *origin, char **result) {
13     // 用sscanf应该也可以, 这里用了strtok
14     char **p = result;
15     *p++ = strtok(origin, " ");
16     while ((*p++ = strtok(NULL, " ")) != NULL) ;
17     // 如果最终返回结果为NULL后这个NULL已经读进去了, 而且p++了, 所以要-1
18     return p - result - 1;
19 }
20
21 void output_one_line(char **p, char **line_start, int temp_len, int m) {
22     if (p == line_start) {
23         printf("%s\n", *p);
24         return;
25     }
26     int extra_space_num = m - temp_len;
27     int min_space = extra_space_num / (p - line_start) + 1;
28     int big_space_num = extra_space_num % (p - line_start);
29     int small_space_num = p - line_start - big_space_num;
30     for (char **word = line_start; word < p; word++, small_space_num--) {
31         printf("%s", *word);
32         for (int i = 0; i < min_space; i++) putchar(' ');
33         if (small_space_num <= 0) putchar(' ');
34     }
35     printf("%s\n", *p);
36 }
37
38 void output_str(char **str, int words_num, int m) {
39     char **p = str, **end = str + words_num, **line_start = str;
40     int temp_len = strlen(*p); // 第一个单词
41     for (p++; p < end; p++) {
42         temp_len += strlen(*p) + 1; // 每个单词和左边至少一个空格
43         if (temp_len < m) continue;
44         if (temp_len > m) temp_len -= strlen(*p--) + 1;
45         output_one_line(p, line_start, temp_len, m);
46         p++;
47         line_start = p;
48         temp_len = strlen(*p); // 第一个单词左边无空格
49     }
}
```

```

50 // 此时p == end, 为NULL
51 // 下面这个情况不可能出现, 所以注释了
52 // if (temp_len > m) { // 这样也可能导致困惑, 但懒得改
53 //     temp_len -= strlen(*p--) + 1;
54 //     output_one_line(p, line_start, temp_len, m);
55 //     p++;
56 //     line_start = p;
57 //     temp_len = strlen(*p); // 第一个单词左边无空格
58 // }
59 if (temp_len > 0) {
60     output_one_line(p - 1, line_start, m, m); // 这样传参可能会导致歧义, 但懒
得改
61 }
62 }
63
64 int main() {
65     int t;
66     scanf("%d", &t);
67     for (int i = 0; i < t; i++) {
68         int m;
69         scanf("%d", &m);
70         char s[N + 2]; // 换行符和末尾'\0'
71         char *words[N + 1]; // NULL可能会存进去
72         getchar(); // 换行符
73         fgets(s, N+2, stdin);
74         s[strlen(s) - 1] = 0; // 换行符去掉
75         int n = read_str(s, words);
76         printf("case #%d:\n", i);
77         output_str(words, n, m);
78     }
79 }
80

```

./1039.字符组合.c

```

1 // 16:08
2 // 16:51
3
4 #include<stdio.h>
5 #include<string.h>
6 #define N 16
7
8 // 返回读取的长度
9 int get_unrepeated_chars(char *origin, char *result) {
10     char *p_result = result;
11     int appear[128] = {0};
12     for (char *p = origin; *p; p++) appear[*p]++;
13     for (int i = 0; i < 128; i++) {
14         if (appear[i] > 0) *p_result++ = i;
15     }
16     *p_result = 0;
17     return p_result - result;
18 }
19

```

```

20 // 递归
21 void output_combinations(const char *prefix, char *s, int len) {
22     // 不用const会报错
23     if (len == 1) {
24         printf("%s", prefix);
25         putchar(*s);
26         putchar('\n');
27     }
28     if (len <= 1) return;
29     printf("%s", prefix);
30     putchar(*s);
31     putchar('\n'); // 只输出当前位
32     // putchar(*s);
33     char new_prefix[N + 1];
34     char appending[] = {*s, 0};
35     strcpy(new_prefix, prefix);
36     strcat(new_prefix, appending); // 不能把char类型的传给char*类型的
37     output_combinations(new_prefix, s + 1, len - 1); // 输出当前位
38     output_combinations(prefix, s + 1, len - 1); // 不输出当前位
39 }
40
41 int main() {
42     int t;
43     scanf("%d", &t);
44     for (int i = 0; i < t; i++) {
45         printf("case #d:\n", i);
46         char unrepeated[N + 1], origin[N + 1];
47         scanf("%s", origin);
48         int len = get_unrepeated_chars(origin, unrepeated);
49         output_combinations("", unrepeated, len);
50     }
51 }
52

```

./1040.字符串消除.c

```

1 // 18:40
2 // 19:59
3 // 20:28
4
5 #include<stdio.h>
6 #include<string.h>
7 #define N 100
8
9 // 返回消除掉的字符串个数
10 int eliminated_once(const char *origin, char *result) {
11     const char *end = origin + strlen(origin);
12     int eliminated_num = 0;
13     char *q = result;
14     for (const char *p = origin; p < end - 1; p++) {
15         if (*p != *(p + 1)) *q++ = *p;
16         else {
17             const char *r = p + 1;
18             while (r < end && *(r - 1) == *r) r++; // 找到不同的

```

```

19         eliminated_num += r - p;
20         p = r - 1; // 下一步到for循环里还要++的, 所以这里-1
21     }
22 }
23 if (*(end - 2) != *(end - 1)) *q++ = *(end - 1); // 最后一个字符
24 *q = 0; // 别忘了结束的0
25 return eliminated_num;
26 }
27
28 // 返回消除掉的字符串个数
29 int get_eliminated_string(const char *origin, char *result) {
30     char temp[N + 1];
31     strcpy(temp, origin);
32     int eliminated_num = 0;
33     int once = 0;
34     while ((once = eliminated_once(temp, result)) > 0) {
35         eliminated_num += once;
36         strcpy(temp, result);
37     }
38     return eliminated_num;
39 }
40
41 // 得到字符串中某个位置的回文子串长度
42 int get_palindrome_num(const char *str, const char *p) {
43     int half = 1; // 朝向一边的偏移量, 算上自身
44     while (p - half >= str && *(p + half) &&
45           *(p - half) == *(p + half)) half++;
46     return 2 * half - 1;
47 }
48
49 // 得到最大回文子串长度, 用这个方法会有情况不对, 暂时还未发现原因
50 // BBABCABCCBCABCCBAABBBBACAAACBCBACCACAAACACBABBAAA
51 // 正确答案是44, 但用这样的方法的结果是43
52 // CACCACAABAABCCCAABCAAAAACCAABCBABCCBABBCCBACCACACCCBBBAAAB
53 // 正确答案是58, 但用这样的方法结果是57
54 int get_max_palindrome_num(const char *str) {
55     int max_num = 0;
56     for (const char *p = str; *p; p++) {
57         int temp;
58         if ((temp = get_palindrome_num(str, p)) > max_num)
59             max_num = temp;
60     }
61     return max_num;
62 }
63
64 // pos是src中的某个字符的地址, 可以是末尾0的地址, 把c插入到pos前方, 可以返回插入后的长度但懒得返回
65 void insert_char(char *dst, const char *src, char c, const char *pos) {
66     for (; *src || *(src - 1); dst++) {
67         if (src == pos) *dst = c, pos = NULL; // 取消pos防止下次循环再运行到这里
68         else *dst = *src++;
69     }
70     *dst = 0;
71 }
72

```



```

73 int get_max_eliminated_num(const char *str) {
74     int max_num = 0;
75     for (const char *p = str; *p || *(p - 1); p++) { // 末尾的*p == 0就代表插入到结尾, 不过这里懒得考虑空字符串的情况
76         char temp[N + 2], temp2[N + 2]; // 插入了一个字符
77         for (char c = 'A'; c <= 'C'; c++) {
78             insert_char(temp, str, c, p);
79             int num = get_eliminated_string(temp, temp2);
80             if (num > max_num) max_num = num;
81         }
82     }
83     return max_num;
84 }
85
86 int main() {
87     int t;
88     char s[N + 1];
89     scanf("%d", &t);
90     for (int i = 0; i < t; i++) {
91         scanf("%s", s);
92         printf("case #%d:\n", i);
93         // char eliminated_string[N + 1];
94         // int eliminated_num = get_eliminated_string(s,
eliminated_string);
95         // int max_substring_len =
get_max_palindrome_num(eliminated_string);
96         // printf("%d\n", eliminated_num + max_substring_len + 1);
97         printf("%d\n", get_max_eliminated_num(s));
98     }
99 }
100

```

./1041.十六进制.c

```

1 // 20:31
2 // 20:48
3 // 20:59
4
5 #include<stdio.h>
6 #include<stdlib.h>
7 #include<string.h>
8 #define N 100000
9
10 int main() {
11     // printf("%lld\n", atoll("0x10")); // 失败
12     // printf("%lld\n", strtoull("0x10", NULL, 16)); // 成功
13     // printf("%lld\n", strtoull("10", NULL, 16)); // 成功
14     // printf("%lld\n", strtoull("0x10g8", NULL, 16)); // 成功
15     // printf("%lld\n", strtoull("ggggg", NULL, 16)); // 结果是0
16     char s[N + 1];
17     scanf("%s", s);
18     char *p = s;
19     int has_num = 0;
20     while ((p = strstr(p, "0x")) != NULL) {

```

```

21 // 可能出现0xggggggggx000x的情况, 提取出的p转成数字是0, 但实际上不应该算上这种
    情况
22     if (strlen(p) > 2 &&
23         (*(p + 2) >= '0' && *(p + 2) <= '9') ||
24         (*(p + 2) >= 'a' && *(p + 2) <= 'f')) {
25         printf("%llu ", strtoull(p, NULL, 16));
26         has_num = 1;
27     }
28     p += 2;
29 }
30 if (!has_num) printf("-1");
31 }
32

```

./1042.字串变换.c

```

1 // 8:05
2 // 9:52
3 // 10:10
4 // 10:50
5
6 #include<stdio.h>
7 #include<stdlib.h>
8 #define N 100
9 #define M 100000
10
11 typedef struct {
12     char c;
13     int times;
14 } str[N];
15 typedef long long ll;
16 str list[M];
17
18 int read_str(const char *s, str dst) {
19     int i = 0;
20     for (const char *p = s; *p; p++) {
21         const char *q = p;
22         for (q++; *q && *p == *q; q++);
23         dst[i].c = *p;
24         dst[i].times = q - p;
25         i++;
26         p = q - 1; // 之后一步要p++
27     }
28     return i;
29 }
30
31 // 共n行, 每行m个不连续相同字符
32 int is_possible(int n, int m) {
33     for (int i = 1; i < n; i++) {
34         for (int j = 0; j < m; j++) {
35             if (list[i][j].c != list[0][j].c) return 0;
36         }
37     }
38     return 1;

```

```

39 }
40
41 // 其中一个字符全部变换到target次数所需要的最小次数
42 ll get_one_col_min_times(int n, int j, int target) {
43     ll result = 0;
44     for (int i = 0; i < n; i++) {
45         result += abs(list[i][j].times - target);
46     }
47     return result;
48 }
49
50 int cmp(const void *pa, const void *pb) {
51     int a = *(int*)pa;
52     int b = *(int*)pb;
53     return a - b;
54 }
55
56 // 无法达到目标时返回-1
57 ll get_min_change_times(int n, int m) {
58     if (!is_possible(n, m)) return -1LL;
59     ll result = 0;
60     for (int j = 0; j < m; j++) {
61         // 不是距离平均数的和最小, 而是中位数
62         int times[M];
63         for (int i = 0; i < n; i++) times[i] = list[i][j].times;
64         qsort(times, n, sizeof(int), cmp);
65         // for (int i = 0; i < n; i++) { // 这样复杂度太高了, 超时
66         //     int less_than_medium = 0;
67         //     medium = list[i][j].times;
68         //     for (int k = 0; k < n; k++)
69         //         if (list[k][j].times < medium) less_than_medium++;
70         //     if (less_than_medium == n / 2) break;
71         //     // avg += (double)list[i][j].times / n;
72         // }
73         int medium = times[n / 2];
74         ll temp1 = get_one_col_min_times(n, j, medium);
75         ll temp2 = get_one_col_min_times(n, j, medium + 1);
76         result += temp1 < temp2 ? temp1 : temp2;
77     }
78     return result;
79 }
80
81 ll read_and_get_times(int n) {
82     int m = -1;
83     for (int i = 0; i < n; i++) {
84         char s[N + 1];
85         scanf("%s", s);
86         int temp = read_str(s, list[i]);
87         if (m < 0) m = temp;
88         else if (m != temp) return -1LL; // 提前排除掉不重复字符数量不同的情况
89     }
90     return get_min_change_times(n, m);
91 }
92
93 int main() {

```

```
94     int n;
95     scanf("%d", &n);
96     printf("%lld", read_and_get_times(n));
97 }
98
```

./1054.负二进制.c

```
1 // 10:00
2 // 11:00
3 // 14:50
4 // 15:01
5 // 15:06
6
7 #include<stdio.h>
8 #include<string.h>
9 #define N 100
10
11 typedef struct {
12     int nums[N]; // 从低位到高位
13     int count;
14     int sign;
15 } BigInt;
16
17 void str_to_BigInt(char *s, BigInt *bigint) {
18     int len = strlen(s);
19     for (int i = len - 1; i >= 1; i--) {
20         bigint->nums[bigint->count++] = s[i] - '0';
21     }
22     if (s[0] == '-')
23         bigint->sign = -1;
24     else
25         bigint->nums[bigint->count++] = s[0] - '0';
26 }
27
28 void carry(BigInt *bigint) {
29     int i = 0;
30     for (; i < N; i++) {
31         if (bigint->nums[i] >= 10) {
32             bigint->nums[i + 1] += bigint->nums[i] / 10;
33             bigint->nums[i] %= 10;
34         }
35     }
36     for (i = N - 1; i >= 0; i--) {
37         if (bigint->nums[i]) break;
38     }
39     bigint->count = i + 1;
40 }
41
42 void add1(BigInt *bigint) {
43     bigint->nums[0] ++;
44     carry(bigint);
45 }
46
```

```

47 // 返回余数
48 int div_2(BigInt *bigint) {
49     int temp = 0;
50     int i;
51     for (i = bigint->count - 1; i > 0; i--) {
52         temp = temp * 10 + bigint->nums[i];
53         if (temp < 2) {
54             bigint->nums[i] = 0;
55             temp = temp * 10 + bigint->nums[--i];
56         }
57         bigint->nums[i] = temp / 2;
58         temp %= 2;
59     }
60     if (i >= 0) {
61         temp = temp * 10 + bigint->nums[0];
62         bigint->nums[0] = temp / 2;
63     }
64     carry(bigint);
65     return temp % 2;
66 }
67
68 // 返回余数
69 int div_neg_2(BigInt *bigint) {
70     int remainder = div_2(bigint);
71     if (bigint->sign < 0 && remainder == 1) {
72         // remainder += 2;
73         add1(bigint);
74     }
75     bigint->sign = - bigint->sign;
76     return remainder;
77 }
78
79 void print_neg_2_base(BigInt *bigint) {
80     char temp[N * 4];
81     int i = 0;
82     if (bigint->count <= 1 && bigint->nums[0] == 0) {
83         putchar('0');
84         return;
85     }
86     while (bigint->count > 1 || bigint->nums[0] > 0) {
87         temp[i++] = div_neg_2(bigint) + '0';
88     }
89     for (i--; i >= 0; i--) {
90         printf("%c", temp[i]);
91     }
92 }
93
94 int main() {
95     BigInt n = {{0}, 0, 1};
96     char s[N + 10];
97     scanf("%s", s);
98     str_to_BigInt(s, &n);
99     print_neg_2_base(&n);
100 }
101

```

./1055.数字排序.c

```
1 // 8:02
2 // 8:29
3
4 #include<stdio.h>
5 #include<stdlib.h>
6
7 typedef struct {
8     double value;
9     char str[101];
10 } num;
11
12 num nums[100];
13
14 int cmp(const void *pa, const void *pb) {
15     num a = *(num*)pa;
16     num b = *(num*)pb;
17     if (a.value == b.value) {
18         return 0;
19     }
20     if (a.value > b.value) {
21         return 1;
22     } else {
23         return -1;
24     }
25 }
26
27 int main() {
28     int n;
29     char str[101];
30     scanf("%d", &n);
31     for (int i = 0; i < n; i++) {
32         scanf("%s", nums[i].str);
33         nums[i].value = atof(nums[i].str);
34     }
35     qsort(nums, n, sizeof(num), cmp);
36     for (int i = 0; i < n; i++) {
37         printf("%s\n", nums[i].str);
38     }
39 }
40
```

./1055.计算多项式的系数.c

```
1 // 10:06
2 // 11:50
3 // 16:10
4 // 16:32
5 #include<stdio.h>
6 #define MOD_NUM 10007
7
```

```

8  /* 计算a*b对MOD_NUM取模的值 */
9  int multiply(int a, int b) {
10     long long temp = a % MOD_NUM;
11     temp *= b;
12     int result = temp % MOD_NUM;
13     return result;
14 }
15
16 /* 计算a的n次方对MOD_NUM取模的值 */
17 int power(int a, int n) {
18     int result = 1;
19     for (; n > 0; n--) { // 循环n次
20         result = multiply(result, a);
21     }
22     return result;
23 }
24
25 /* 计算C(k, n)的组合数对MOD_NUM取模的值 */
26 //int combination(int k, int n) {
27 //    int i = .
28 //    int ai = 1; // 每一项乘起来
29 //    int ai = multiply(1, k); // 从k乘到k-n+1, 共n个
30 //    int result = 1;
31 //    int denominator = multiply(1, n); // 从n乘到1
32 //    for (int i = 1; i <= n; i++) { // 循环n-1次
33 //        int remainder = k % i;
34 //        int ai = k - remainder;
35 //        if (remainder == 0) {
36 //            ai /= n;
37 //        } else {
38 //            ai /= remainder;
39 //        }
40 //        result = multiply(result, ai);
41 //        denominator = multiply(denominator, n);
42 //    }
43 //    int result = numerator / denominator;
44 //    if (result == 0) {
45 //        result = numerator;
46 //    }
47 //    return result;
48 //}
49
50 int combinations[1001][1001] = {{0}};
51 void init_combination_triangle() {
52     combinations[0][0] = 1;
53     for (int i = 1; i < 1001; i++) {
54         combinations[i][0] = 1;
55         for (int j = 1; j < i; j++) {
56             combinations[i][j] = combinations[i-1][j-1] + combinations[i-1]
[j];
57             combinations[i][j] %= MOD_NUM;
58         }
59         combinations[i][i] = 1;
60     }
61 }

```

```

62
63 int main() {
64     int t;
65     int a, b, k, n, m;
66     init_combination_triangle();
67     scanf("%d", &t);
68     for (int i = 0; i < t; i++) {
69         scanf("%d%d%d%d", &a, &b, &k, &n, &m);
70         int an = power(a, n);
71         int bm = power(b, m);
72         int comb_k_n = combinations[k][n];
73         int result = multiply(an, bm);
74         result = multiply(result, comb_k_n);
75         printf("case #d:\n%d\n", i, result);
76     }
77     return 0;
78 }
79
80

```

./1056.找出最小字符串.c

```

1 // 9:10
2
3 #include<stdio.h>
4 #define N 100
5
6 typedef struct {
7     int alpha;
8     int times;
9 } Repeat;
10 Repeat alphas[N];
11
12 int main() {
13     int c;
14     int i = 0;
15     while ((c = getchar()) != '\n') {
16         alphas[i].alpha = c;
17         alphas[i].times = 1;
18         i++;
19     }
20     for (int j = 0; j < i - 1; j++) {
21         int k = j + 1;
22         while (alphas[j].alpha == alphas[k].alpha)
23             k++;
24         if (alphas[j].alpha < alphas[k].alpha) {
25             for (int i = j; i < k; i++) {
26                 alphas[i].times = 2;
27             }
28         }
29     }
30     for (int j = 0; j < i; j++) {
31         for (int k = 0; k < alphas[j].times; k++) {
32             putchar(alphas[j].alpha);

```



```

33     }
34 }
35 }
36
37

```

./1056.浮点数加法.c

```

1 // 19:06
2 // 19:09
3 // 19:36
4 // 09:07
5 // 09:53
6 // 10:17
7
8 #include<stdio.h>
9 #include<string.h>
10 #define MAX_LEN 510
11
12 typedef struct {
13     int only_nums[MAX_LEN * 2]; // 前半部分整数, 后半部分小数
14     int point_pos_from_little;
15 } Number;
16
17 void read_number(char str[], int len, Number *num) {
18     int i = 0;
19     int temp[MAX_LEN];
20     for (; i < len; i++) {
21         if (str[i] == '.') {
22             num->point_pos_from_little = len - 1 - i;
23             break;
24         }
25         temp[i] = str[i] - '0';
26     }
27     temp[i] = 0; // 此时i是temp的长度
28     memset(num->only_nums, 0, (MAX_LEN-i)*sizeof(int)); // 整数前面都是0
29     memcpy(&num->only_nums[MAX_LEN-i], temp, i*sizeof(int)); // 放到整数里
30     i++; // 跳过这个小数点
31     for (int j = 0; i < len; i++, j++) {
32         num->only_nums[MAX_LEN + j] = str[i] - '0';
33     }
34     memset(num->only_nums+MAX_LEN+i, 0, (MAX_LEN-i)*sizeof(int));
35 }
36
37 void carry(Number *num) {
38     for (int i = MAX_LEN * 2 - 1; i >= 0; i--) {
39         if (num->only_nums[i] >= 10) {
40             num->only_nums[i-1] += num->only_nums[i] / 10;
41             num->only_nums[i] %= 10;
42         }
43     }
44 }
45
46 void add(Number *a, Number *b, Number *result) {

```

```

47     memset(result->only_nums, 0, MAX_LEN*2*sizeof(int));
48     for (int i = MAX_LEN*2 - 1; i>=0; i--) {
49         result->only_nums[i] += a->only_nums[i] + b->only_nums[i];
50     }
51     carry(result);
52 }
53
54 void print_number(Number *num, int n) {
55     if (num->only_nums[MAX_LEN+n] >= 5) {
56         num->only_nums[MAX_LEN+n-1] += 1;
57         carry(num);
58     }
59     int i = 0;
60     while(i < MAX_LEN && num->only_nums[i] == 0) i++;
61     if (i >= MAX_LEN) {
62         putchar('0');
63     } else {
64         while (i < MAX_LEN) {
65             putchar(num->only_nums[i] + '0');
66             i++;
67         }
68     }
69     putchar('.');
70     for (int j = 0; j < n; j++, i++) {
71         putchar(num->only_nums[i] + '0');
72     }
73 }
74
75 int main() {
76     Number a={{0},0}, b={{0},0}, result={{0},0};
77     char str_a[MAX_LEN], str_b[MAX_LEN];
78     int n;
79     scanf("%s %s %d", str_a, str_b, &n);
80     read_number(str_a, strlen(str_a), &a);
81     read_number(str_b, strlen(str_b), &b);
82     add(&a, &b, &result);
83     print_number(&result, n);
84 }
85
86

```

./1057.整数分解.c

```

1 // 9:20
2 // 10:29
3
4 #include<stdio.h>
5 #define PRIME_NUM 168
6 #define MAX_N 1000
7

```

```

8  int primes[PRIME_NUM] =
    {2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,89,97,101,
    103,107,109,113,127,131,137,139,149,151,157,163,167,173,179,181,191,193,197,
    199,211,223,227,229,233,239,241,251,257,263,269,271,277,281,283,293,307,311,
    313,317,331,337,347,349,353,359,367,373,379,383,389,397,401,409,419,421,431,
    433,439,443,449,457,461,463,467,479,487,491,499,503,509,521,523,541,547,557,
    563,569,571,577,587,593,599,601,607,613,617,619,631,641,643,647,653,659,661,
    673,677,683,691,701,709,719,727,733,739,743,751,757,761,769,773,787,797,809,
    811,821,823,827,829,839,853,857,859,863,877,881,883,887,907,911,919,929,937,
    941,947,953,967,971,977,983,991,997};
9  long long plans[MAX_N][PRIME_NUM] = {{0}};
10
11  // n在只能用从0到prime_num个数字时的分解方法
12  long long get_one_plan(int n, int prime_num) {
13      long long result = plans[n][prime_num - 1];
14      // for (int i = 0; i < PRIME_NUM; i++) {
15          //     if (primes[i] == n) {
16              //         result ++;
17              //         break;
18          //     }
19      // }
20      while (n >= primes[prime_num]) {
21          n -= primes[prime_num];
22          result += plans[n][prime_num - 1];
23      }
24      if (n == 0) {
25          result ++;
26      }
27      return result;
28  }
29
30  void get_plans() {
31      for (int j = 0; j < PRIME_NUM; j++) {
32          // plans[0][j] = 0;
33          plans[2][j] = 1;
34          plans[3][j] = 1;
35      }
36      for (int i = 3; i < MAX_N; i++) {
37          if (i % 2 == 0) {
38              plans[i][0] = 1;
39          } else {
40              plans[i][0] = 0;
41          }
42      }
43      for (int i = 4; i < MAX_N; i++) {
44          for (int j = 1; j < PRIME_NUM; j++) {
45              plans[i][j] = get_one_plan(i, j);
46          }
47      }
48  }
49
50  int main() {
51      int n;
52      get_plans();
53      scanf("%d", &n);

```

```
54     printf("%lld", plans[n][PRIME_NUM - 1]);
55 }
56
```

./1057.计算a的n次方的大整数.c

```
1 // 20:11
2 // 20:17
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <string.h>
6 #define MAX 200
7
8 typedef struct
9 {
10     int len;
11     int s[MAX+1];
12 } hp;
13
14 void input(hp *a, int x) //读入数字
15 {
16     int i;
17
18     a->len = 0;
19
20     while (x > 0)
21     {
22         a->s[1 + a->len++] = x % 10;
23         x /= 10;
24     }
25
26     for (i = a->len + 1; i <= MAX; i++)
27         a->s[i] = 0;
28 }
29
30 void print(hp *y) //打印数字
31 {
32     int i;
33     for (i = y->len; i >= 1; i--)
34         printf("%d", y->s[i]);
35     printf("\n");
36 }
37
38 void multiplyh(hp *a, hp *b, hp *c) //高精度 * 高精度
39 {
40     int i, j, len;
41
42     for (i = 1; i <= MAX; i++) c->s[i] = 0;
43
44     for (i = 1; i <= a->len; i++)
45     {
46         for (j = 1; j <= b->len; j++)
47         {
48             c->s[i+j-1] += a->s[i] * b->s[j];
```

```

49         c->s[i+j] += c->s[i+j-1] / 10;
50         c->s[i+j-1] %= 10;
51     }
52 }
53
54 len = a->len + b->len + 1;
55 while (len > 1 && c->s[len] == 0) len--;
56 c->len = len;
57 }
58
59 void power(hp *a, int b, hp *c) //高精度乘方  $c = a^b$ 
60 {
61     hp e;
62
63     if (b == 0)
64     {
65         c->len = 1;
66         c->s[1] = 1;
67     }
68     else if (b == 1)
69     {
70         memcpy(c, a, sizeof(hp));
71     }
72     else
73     {
74         power(a, b / 2, &e);
75         multiplyh(&e, &e, c);
76
77         if (b % 2 == 1)
78         {
79             memcpy(&e, c, sizeof(hp));
80             multiplyh(&e, a, c);
81         }
82     }
83 }
84
85 int main() {
86     int t;
87     scanf("%d", &t);
88     int a, n;
89     for (int i = 0; i < t; i++) {
90         scanf("%d%d", &a, &n);
91         hp a_hp, result;
92         input(&a_hp, a);
93         power(&a_hp, n, &result);
94         printf("case #%d:\n", i);
95         print(&result);
96     }
97     return 0;
98 }
99

```

./1058.一个游戏.c

```

1 // 20:18
2 // 20:27
3 // 20:36
4
5 #include<stdio.h>
6 #include<string.h>
7
8 int main() {
9     int t;
10    scanf("%d", &t);
11    for (int i = 0; i < t; i++) {
12        int n;
13        int head[100];
14        int tail[100];
15        int a_to_b[100][100];
16        scanf("%d", &n);
17        memset(head, 0, 100*sizeof(int));
18        memset(tail, 0, 100*sizeof(int));
19        memset(a_to_b, 0, 100*100*sizeof(int));
20        int justify = 1;
21        for (int j = 0; j < n; j++) {
22            int ai, bi;
23            scanf("%d%d", &ai, &bi);
24            head[ai] ++;
25            tail[bi] ++;
26            if (a_to_b[ai][bi]) { // 去重
27                continue;
28            } else {
29                a_to_b[ai][bi] = 1;
30            }
31            if (head[ai] > 1 || head[ai] > 0 && tail[ai] > 0
32                || head[bi] > 0 && tail[bi] > 0) {
33                justify = 0;
34            }
35        }
36        if (justify) {
37            printf("Lucky dxw!\n");
38        } else {
39            printf("Poor dxw!\n");
40        }
41    }
42 }
43

```

./1059.计算a的n次方的大整数.c

```

1 // 10:14
2 // 10:49
3
4 #include<stdio.h>
5 #include<string.h>
6 #define N 1005
7
8 typedef struct {

```

```

9     int num[N]; // 从低位到高位
10    int cnt; // 有效数字位数
11 } BigInt;
12
13 void init_num(BigInt *num) {
14     memset(num->num, 0, N * sizeof(int));
15     num->cnt = 0;
16 }
17
18 void read_num(int a, BigInt *result) {
19     init_num(result);
20     result->num[0] = a;
21     result->cnt = 1;
22 }
23
24 void output_num(BigInt *num) {
25     for (int i = num->cnt - 1; i >= 0; i--) {
26         putchar(num->num[i] + '0');
27     }
28     putchar('\n');
29 }
30
31 void carry(BigInt *num) {
32     int i = 0;
33     for (; i < num->cnt; i++) {
34         if (num->num[i] >= 10) {
35             num->num[i + 1] += num->num[i] / 10;
36             num->num[i] %= 10;
37         }
38     }
39     for (; i < N; i++) {
40         if (num->num[i] > 0) {
41             num->cnt = i + 1;
42         }
43         if (num->num[i] >= 10) {
44             num->num[i + 1] += num->num[i] / 10;
45             num->num[i] %= 10;
46         }
47     }
48 }
49
50 void cpy_num(BigInt *dst, BigInt *src) {
51     memcpy(dst, src, sizeof(BigInt));
52 }
53
54 void multiply(BigInt *a, BigInt *b, BigInt *result) {
55     init_num(result);
56     // int weight = 1;
57     for (int i = 0; i < a->cnt; i++) {
58         for (int j = 0; j < b->cnt; j++) {
59             result->num[i + j] += a->num[i] * b->num[j];
60         }
61         // weight *= 10;
62     }
63     carry(result);

```

```

64 }
65
66 void pow(BigInt *a, int n, BigInt *result) {
67     // init_num(result);
68     BigInt temp;
69     read_num(1, result);
70     for (int i = 0; i < n; i++) {
71         multiply(a, result, &temp);
72         cpy_num(result, &temp);
73     }
74 }
75
76 int main() {
77     int t;
78     int a, n;
79     BigInt big_a, result;
80     scanf("%d", &t);
81     for (int i = 0; i < t; i++) {
82         printf("case #%d:\n", i);
83         scanf("%d%d", &a, &n);
84         read_num(a, &big_a);
85         pow(&big_a, n, &result);
86         output_num(&result);
87     }
88 }
89

```

./1061.计算n! 右端0的个数(II).c

```

1 // 8:06
2 // 8:23
3 // 8:44
4
5 #include<stdio.h>
6
7 int get_5_factor_nums_in_factorial(int num) {
8     int result = 0;
9     for (int i = 1; i <= num; i++) {
10         int temp = i;
11         while (temp % 5 == 0) {
12             temp /= 5;
13             result++;
14         }
15     }
16     return result;
17 }
18
19 int main() {
20     int t;
21     scanf("%d", &t);
22     for (int i = 0; i < t; i++) {
23         int n;
24         scanf("%d", &n);
25         printf("case #%d:\n", i);

```



```
26     printf("%d\n", get_5_factor_nums_in_factorial(n));
27     }
28 }
29
```

./1062.计算2的N次方.c

```
1 // 8:59
2 // 9:02
3
4 #include<stdio.h>
5
6 int main() {
7     int t;
8     scanf("%d", &t);
9     for (int i = 0; i < t; i++) {
10         int n;
11         scanf("%d", &n);
12         printf("case #d:\n%d\n", i, 1<<n);
13     }
14 }
15
```

./1063.二进制倒置.c

```
1 // 9:04
2 // 9:15
3 // 9:36
4 // 10:01
5
6 #include<stdio.h>
7 #include<string.h>
8 #define N 101
9 #define M 334
10
11 // 倒置
12 typedef struct {
13     int s[N];
14     int cnt;
15 } Big;
16
17 void init_big(Big *num) {
18     memset(num, 0, sizeof(Big));
19     // num->cnt = 1; // 初始化为0的时候是0位数字
20 }
21
22 void read(char s[], Big *num) {
23     init_big(num);
24     for (int i = strlen(s) - 1; i >= 0; i--) {
25         num->s[num->cnt++] = s[i] - '0';
26     }
27 }
```

```

28
29 void output(Big *num) {
30     for (int *p = num->s + num->cnt - 1; p >= num->s; p--) {
31         putchar(*p + '0');
32     }
33     putchar('\n');
34 }
35
36 // 顺便更新num->cnt
37 void carry(Big *num) {
38     for (int i = 0; i < N - 1; i++) {
39         if (num->s[i] >= 10) {
40             num->s[i + 1] += num->s[i] / 10;
41             num->s[i] %= 10;
42         }
43     }
44     int i = N - 1;
45     for (; i > 0; i--) {
46         if (num->s[i] != 0) break;;
47     }
48     num->cnt = i + 1; // 如果数字为0, cnt是1
49 }
50
51 // 返回余数
52 int divide_by_2(Big *num) {
53     int borrow = 0;
54     for (int i = num->cnt - 1; i > 0; i--) {
55         borrow += num->s[i];
56         if (borrow < 2){
57             num->s[i] = 0;
58             borrow *= 10;
59             continue;
60         }
61         num->s[i] = borrow >> 1;
62         borrow &= 1;
63         borrow *= 10;
64     }
65     borrow += num->s[0];
66     num->s[0] = borrow >> 1;
67     borrow &= 1;
68     carry(num);
69     return borrow;
70 }
71
72 // 返回二进制位数
73 int read_to_binary(Big *num, int binary[]) {
74     int *p = binary;
75     for (; num->cnt != 1 || num->s[0] != 0; p++) {
76         *p = divide_by_2(num);
77     }
78     return p - binary;
79 }
80
81 void multiply_2(Big *num) {
82     for (int i = 0; i < num->cnt; i++) num->s[i] <<= 1;

```

```

83     carry(num);
84 }
85
86 void add_1(Big *num) {
87     num->s[0] += 1;
88     carry(num);
89 }
90
91 void binary_to_big(int binary[], int binary_len, Big *num) {
92     init_big(num);
93     num->cnt = 1; // 初始化后作为1处理
94     for (int i = 0; i < binary_len - 1; i++) {
95         if (binary[i]) add_1(num);
96         multiply_2(num);
97     }
98     if (binary[binary_len - 1]) add_1(num);
99 }
100
101 int main() {
102     int t;
103     scanf("%d", &t);
104     for (int i = 0; i < t; i++) {
105         char s[N + 1];
106         scanf("%s", &s);
107         Big n;
108         read(s, &n);
109         int binary[M];
110         int binary_len = read_to_binary(&n, binary);
111         binary_to_big(binary, binary_len, &n);
112         printf("case #%d:\n", i);
113         output(&n);
114     }
115 }
116

```

./1064.A-B(Big Integer).c

```

1 // 10:11
2
3 #include<stdio.h>
4 #include<string.h>
5 #define N 500
6
7 // 倒置
8 typedef struct {
9     int s[N];
10    int cnt;
11    int sign;
12 } Big;
13
14 void init(Big *num) {
15     memset(num, 0, sizeof(Big));
16     num->sign = 1;
17 }

```

```

18
19 void input(Big *num, char s[]) {
20     init(num);
21     for (char *p = s + strlen(s) - 1; p >= s; p--) {
22         num->s[num->cnt++] = *p - '0';
23     }
24 }
25
26 void output(Big *num) {
27     if (num->sign < 0) putchar('-');
28     for (int i = num->cnt - 1; i > 0; i--) {
29         putchar(num->s[i]);
30     }
31     putchar('\n');
32 }
33
34 void carry(Big *num) {
35     for (int i = 0; i < N - 1; i++) {
36         if (num->s[i] >= 10) {
37             num->s[i + 1] = num->s[i] / 10;
38             num->s[i] /= 10;
39         }
40     }
41     int i = N - 1;
42     for (; i > 0; i--) {
43         if (num->s[i] != 0) break;
44     }
45     num->cnt = i;
46 }
47
48 int cmp(Big *a, Big *b) {
49     if (a->cnt != b->cnt) return a->cnt - b->cnt;
50     for (int i = a->cnt - 1; i >= 0; i--) {
51         if (a->s[i] == b->s[i]) continue;
52         if (a->s[i] > b->s[i]) return 1;
53         else return -1;
54     }
55     return 0;
56 }
57
58 void minus(Big *a, Big *b, Big *result) {
59     init(result);
60     int borrow = 0;
61     for (int i = N - 1; i >= 0; i++) {
62         borrow += a->s[i];
63         if (borrow < b->s[i]) {
64             result->s[i + 1]--;
65             borrow += 10;
66         }
67         result->s[i] = borrow - b->s[i];
68         borrow = 0;
69     }
70     carry(result);
71 }
72

```

```

73 int main() {
74     char a[N + 1], b[N + 1];
75     Big A, B, result;
76     while (scanf("%s", a) != EOF) {
77         scanf("%s", b);
78         input(&A, a);
79         input(&B, b);
80         if (cmp(&A, &B) < 0) {
81             minus(&B, &A, &result);
82             result.sign = -1;
83         }
84         minus(&A, &B, &result);
85         output(&result);
86     }
87 }
88

```

./1065.浮点数减法.c

```

1 // 10:34
2 // 11:55
3
4 // 13:32
5 // 14:53
6
7 #include<stdio.h>
8 #include<string.h>
9 #define N 600
10 #define M 50
11
12 typedef struct {
13     int s[2 * N];
14     int cnt; // 最右边的非零数字的位置
15     int sign;
16 } BigFloat;
17 // 前N位是小数, 后N位是整数
18
19 void init(BigFloat *num) {
20     memset(num, 0, sizeof(BigFloat));
21     num->sign = 1;
22     // num->cnt = 1;
23 }
24
25 void read_num(BigFloat *num, char *s) {
26     init(num);
27     if (*s == '-') {
28         num->sign = -1;
29         s++;
30     }
31     char *p_point = strchr(s, '.');
32     char *end = s + strlen(s);
33     int *p_result = num->s + N;
34     if (p_point == NULL) p_point = end;
35     for (char *p = p_point + 1; p < end; p++) {

```

```

36     *--p_result = *p - '0';
37 }
38 p_result = num->s + N;
39 for (char *p = p_point - 1; p >= s; p--) {
40     *p_result++ = *p - '0';
41 }
42 // num->cnt = p - num->s - N;
43 }
44
45 int cmp(BigFloat *a, BigFloat *b) {
46     for (int i = 2 * N - 1; i >= 0; i--) {
47         if (a->s[i] == b->s[i]) continue;
48         return a->s[i] - b->s[i];
49     }
50     return 0;
51 }
52
53 void carry(BigFloat *num, int n) {
54     for (num->cnt = 2 * N - 1; num->cnt > 0; num->cnt--) {
55         if (num->s[num->cnt] != 0) break;
56     }
57     for (int i = 0; i <= num->cnt; i++) {
58         if (num->s[i] < 0) {
59             num->s[i] += 10;
60             num->s[i + 1] --;
61         }
62         if (num->s[i] >= 10) {
63             num->s[i + 1] += num->s[i] / 10;
64             num->s[i] %= 10;
65         }
66     }
67     for (num->cnt = 2 * N - 1; num->cnt > 0; num->cnt--) {
68         if (num->s[num->cnt] != 0) break;
69     }
70     if (num->s[num->cnt] < 0) {
71         for (int i = 0; i < 2 * N; i++)
72             num->s[i] *= -1;
73         num->sign = -1;
74         carry(num, n);
75     }
76 }
77
78 void minus(BigFloat *a, BigFloat *b, BigFloat *result, int n) {
79     init(result);
80     for (int i = 0; i < 2 * N; i++) {
81         result->s[i] = a->s[i] - b->s[i];
82     }
83     // if (result->s[N - n - 1] >= 5) result->s[N - n]++;
84     carry(result, n);
85     if (result->s[N - n - 1] >= 5) {
86         result->s[N - n]++;
87         carry(result, n);
88     }
89 }
90

```

```

91 //void minus_with_cmp(BigFloat *a, BigFloat *b, BigFloat *result) {
92 //
93 //}
94
95 void output(BigFloat *num, int n){
96     int *p = num->s;
97     if (num->sign < 0) {
98         printf("-");
99         p++;
100    }
101    int i = 2 * N - 1;
102    for (; i > N; i--) {
103        if (num->s[i] != 0) break;
104    }
105    for (; i >= N; i--) {
106        putchar(num->s[i] + '0');
107    }
108    putchar('.');
109    // int end_0 = 0;
110    // for (; end_0 < N; end_0++) {
111    //     if (num->s[end_0]) break;
112    // }
113    for (int i = N - 1; i >= N - n; i--) {
114        putchar(num->s[i] + '0');
115    }
116    putchar('\n');
117 }
118
119 int main() {
120     char s[N + 1];
121     BigFloat a, b, result;
122     scanf("%s", s);
123     read_num(&a, s);
124     scanf("%s", s);
125     read_num(&b, s);
126     int n;
127     scanf("%d", &n);
128     minus(&a, &b, &result, n);
129     output(&result, n);
130     return 0;
131 }
132
133

```

./1081.种田.c

```

1 // 9:28
2 // 9:44
3
4 #include<stdio.h>
5
6 // long long body_power[;]
7
8 long long get_body_power(long long x, long long y) {

```

```

 9     long long max_square_num = y / x;
10     long long result = 4 * x * max_square_num;
11     y %= x;
12     if (y <= 0)
13         return result;
14     // if (y > x)
15     //     return result + get_body_power(x, y);
16     // else
17     return result + get_body_power(y, x);
18 }
19
20 int main() {
21     long long x, y;
22     scanf("%lld %lld", &x, &y);
23     printf("%lld\n", get_body_power(y, x));
24     return 0;
25 }
26

```

./1082.波兰表达式.c

```

 1 // 9:47
 2 // 10:20
 3
 4 #include <stdio.h>
 5 #include <string.h>
 6 #include <stdlib.h>
 7 #include <ctype.h>
 8
 9 char *current_pos;
10 char expression[151];
11
12 double calculate(char *str) {
13     char *operator_;
14     operator_ = strtok(str, " ");
15     current_pos = operator_ + strlen(operator_) + 1;
16     if (strlen(operator_) > 1 || isdigit(*operator_)) { // 说明是个数字
17         return atof(operator_);
18     }
19     double operand1 = calculate(current_pos);
20     double operand2 = calculate(current_pos);
21     switch (*operator_) {
22         case '+':
23             return operand1 + operand2;
24         case '-':
25             return operand1 - operand2;
26         case '*':
27             return operand1 * operand2;
28         case '/':
29             return operand1 / operand2;
30     }
31 }
32
33 int main(int argc, char *argv[]) {

```



```

34     int t;
35     scanf("%d", &t);
36     getchar(); // 去掉换行
37     for (int i = 0; i < t; i++) {
38         fgets(expression, 150, stdin);
39         printf("case #d:\n%.6lf\n", i, calculate(expression));
40     }
41     return 0;
42 }
43

```

./1083.地铁站.c

```

1 // 10:26
2 // 10:53
3
4 #include <stdio.h>
5
6 long long board_people[21] = {0}, unboard_people[21] = {0},
7 people_boards[21] = {0};
8 // t是第二站上下车的人数
9 long long get_last_unboard_people(int a, int t, int n) {
10     board_people[1] = a;
11     people_boards[1] = a;
12     board_people[2] = t;
13     unboard_people[2] = t;
14     people_boards[2] = a;
15     for (int i = 3; i < n; i++) {
16         board_people[i] = board_people[i - 1] + board_people[i - 2];
17         unboard_people[i] = board_people[i - 1];
18         people_boards[i] = people_boards[i - 1] + board_people[i] -
19 unboard_people[i];
20     }
21     board_people[n] = 0;
22     unboard_people[n] = people_boards[n - 1];
23     people_boards[n] = 0;
24     return unboard_people[n];
25 }
26
27 int main() {
28     int a, n, m, x;
29     scanf("%d %d %d %d", &a, &n, &m, &x);
30     for (int t = 0; t < m; t++) { // t从1开始的话最后一个测试用例就不对。
31         // 说好的第2站有人上下车呢？还能有0人上下车的？
32         if (m == get_last_unboard_people(a, t, n))
33             break;
34     }
35     printf("%lld\n", people_boards[x]);
36 }

```

./1084.Fj&haozi.c

```
1 // 8:09
2 // 8:50
3
4 #include<stdio.h>
5 #include<string.h>
6 #define MAX_N 101 // 原来写的21导致出错
7
8 long long previous_num[MAX_N] = {0};
9 long long current_num[MAX_N] = {0};
10
11 void expand_num(int n, int m) {
12     for (int i = 0; i < m; i++) {
13         for (int j = 2; j < n; j++) {
14             current_num[j] = previous_num[j - 1] + previous_num[j + 1];
15         }
16         current_num[1] = previous_num[2];
17         current_num[n] = previous_num[n - 1];
18         memcpy(previous_num, current_num, MAX_N * sizeof(long long));
19         memset(current_num, 0, MAX_N * sizeof(long long));
20     }
21 }
22
23 int main() {
24     int n, m, p, t;
25     int cas;
26     scanf("%d", &cas);
27     while(cas--) {
28         scanf("%d %d %d %d", &n, &p, &m, &t);
29         memset(previous_num, 0, MAX_N * sizeof(long long));
30         memset(current_num, 0, MAX_N * sizeof(long long));
31         previous_num[p] = 1;
32         expand_num(n, m);
33         printf("%lld\n", previous_num[t]);
34     }
35 }
36
37
```

./1085.泰波那契数列的前74项.c

```
1 // 8:14
2 // 9:23
3
4 #include<stdio.h>
5
6 long long tribonacci[74] = {0LL, 1LL, 1LL};
7
8 int main(){
9     for(int i = 3; i < 74; i++){
```

```

10     tribonacci[i] = tribonacci[i - 1] + tribonacci[i - 2] + tribonacci[i
- 3];
11     }
12     int t;
13     scanf("%d",&t);
14     for (int i = 0; i < t; i++) {
15         int n;
16         scanf("%d", &n);
17         printf("case #d:\n%lld\n", i, tribonacci[n]);
18     }
19     return 0;
20 }
21

```

./1087.统计字符串个数.c

```

1 // 9:41
2 // 10:01
3
4 #include <stdio.h>
5 #include <string.h>
6 #define substring_width 3
7
8 int end_with_0[substring_width];
9 int end_with_1[substring_width];
10
11 void init() {
12     memset(end_with_0, 0, substring_width * sizeof(end_with_0[0]));
13     memset(end_with_1, 0, substring_width * sizeof(end_with_1[0]));
14 }
15
16 void expand(int n) {
17     end_with_0[0] = 0;
18     end_with_1[0] = 0;
19     end_with_0[1] = 1;
20     end_with_1[1] = 1;
21     for (int i = 2; i < n + 1; i++) {
22         end_with_0[2] = end_with_0[1] + end_with_1[1];
23         end_with_1[2] = end_with_0[2] - end_with_1[0];
24         memmove(end_with_0, end_with_0 + 1, substring_width *
sizeof(end_with_0[0]));
25         memmove(end_with_1, end_with_1 + 1, substring_width *
sizeof(end_with_1[0]));
26     }
27 }
28
29 int main() {
30     int n = 0;
31     while (1) {
32         scanf("%d", &n);
33         if (n < 0)
34             break;
35         init();
36         expand(n);

```

```

37     printf("%d\n", end_with_0[1] + end_with_1[1]);
38     }
39     return 0;
40 }
41

```

./1090.最短路径.c

```

1 // 9:19
2 // 9:39
3
4 #include<stdio.h>
5 #define MAX_N 200
6
7 int matrix[MAX_N][MAX_N];
8 int loss[MAX_N][MAX_N];
9 char route[MAX_N][MAX_N];
10
11 void expand_route(int m, int n) {
12     loss[0][0] = matrix[0][0];
13     for (int i = 1; i < m; i++) {
14         loss[i][0] = loss[i - 1][0] + matrix[i][0];
15         route[i][0] = 'D';
16     }
17     for (int i = 0; i < n; i++) {
18         loss[0][i] = loss[0][i - 1] + matrix[0][i];
19         route[0][i] = 'R';
20     }
21     for (int i = 1; i < m; i++) {
22         for (int j = 1; j < n; j++) {
23             if (loss[i][j - 1] < loss[i - 1][j]) {
24                 loss[i][j] = loss[i][j - 1] + matrix[i][j];
25                 route[i][j] = 'R';
26             }
27             else {
28                 loss[i][j] = loss[i - 1][j] + matrix[i][j];
29                 route[i][j] = 'D';
30             }
31         }
32     }
33 }
34
35 void output_route(int i, int j) {
36     if (i == 0 & j == 0) {
37         return;
38     }
39     if (route[i][j] == 'R') {
40         output_route(i, j - 1);
41         putchar(route[i][j]);
42     } else {
43         output_route(i - 1, j);
44         putchar(route[i][j]);
45     }
46 }

```

```

47
48 int main() {
49     int m, n;
50     scanf("%d%d", &m, &n);
51     for (int i = 0; i < m; i++) {
52         for (int j = 0; j < n; j++) {
53             scanf("%d", &matrix[i][j]);
54         }
55     }
56     expand_route(m, n);
57     printf("%d\n", loss[m - 1][n - 1]);
58     output_route(m - 1, n - 1);
59 }
60

```

./1093.波浪图.c

```

1 // 15:00
2 // 15:23
3
4 #include<stdio.h>
5 #include<string.h>
6 #define N 80
7
8 typedef char Picture[2 * N][2 * N];
9
10 void get_pic(Picture pic, char *s) {
11     memset(pic, ' ', sizeof(Picture));
12     int row = N, col = 0;
13     pic[row][col] = *s;
14     for (s++; *s; s++) {
15         col++;
16         if (*s > *(s - 1)) row--;
17         if (*s < *(s - 1)) row++;
18         pic[row][col] = *s;
19     }
20 }
21
22
23 void output(Picture pic, int len) {
24     char blank[2 * N];
25     memset(blank, ' ', sizeof(blank));
26     int start_row = 0, end_row = 2 * N - 1;
27     for (; memcmp(pic[start_row], blank, sizeof(blank)) == 0; start_row++);
28     for (; memcmp(pic[end_row], blank, sizeof(blank)) == 0; end_row--);
29     for (int i = start_row; i <= end_row; i++) {
30         for (int j = 0; j < len; j++) {
31             putchar(pic[i][j]);
32         }
33         putchar('\n');
34     }
35 }
36
37 int main() {

```

```

38     char s[N + 1];
39     Picture pic;
40     while (scanf("%s", s) != EOF) {
41         get_pic(pic, s);
42         output(pic, strlen(s));
43     }
44 }
45
46

```

./1094.坏掉的彩灯.c

```

1 // 15:07
2 // 16:19
3
4 // 18:38
5 // 18:51
6
7 // 19:18
8 // 19:19
9
10 #include<stdio.h>
11 #include<string.h>
12 #define N 100
13
14 char Colour[] = "RBYG";
15 char Index[128];
16
17 void init() {
18     Index['R'] = 0;
19     Index['B'] = 1;
20     Index['Y'] = 2;
21     Index['G'] = 3;
22 }
23
24 int global_valid = 0;
25
26 int get_possible_solution(char s[], char *result, int len) {
27     // int len = strlen(s);
28     int i = 0;
29     for (; i < len; i++) {
30         if (s[i] == '!') break;
31     }
32     // printf("%s\n%d %d\n", result, i, len);
33     if (i == len) {
34         global_valid = 1;
35         memcpy(result, s, (N + 1) * sizeof(char));
36         return 1;
37     }
38     int possible[] = {1, 1, 1, 1}; // RBYG
39     for (int j = i - 3; j <= i + 3; j++) {
40         if (j >= 0 && j < len && s[j] != '!') {
41             possible[Index[s[j]]] = 0;
42         }

```

```

43     }
44     int valid = 0;
45     for (int j = 0; j < 4; j++)
46         if (possible[j]) valid = 1;
47     if (valid == 0) {
48 //         memcpy(result, s, N); // 还原
49         return 0; // 表明这种情况不可能
50     }
51     for (int j = 0; j < 4; j++) {
52         if (possible[j]) {
53             char temp[N + 1];
54             memcpy(result, s, (N + 1) * sizeof(char));
55             result[i] = Colour[j];
56             memcpy(temp, result, (N + 1) * sizeof(char));
57             global_valid = get_possible_solution(temp, result, len);
58             if (global_valid) return 1;
59         }
60     }
61     return 0;
62 }
63
64 int get_broken_nums(char good[], char broken[], int colors[], int len) {
65 //     int len = strlen(broken);
66     for (int i = 0; i < len; i++) {
67         if (broken[i] == '!') {
68             colors[Index[good[i]]] += 1;
69         }
70     }
71 }
72
73 int main() {
74     init();
75     int t;
76     scanf("%d", &t);
77     for (int i = 0; i < t; i++) {
78         char s[N + 1], result[N + 1];
79         scanf("%s", s);
80         int len = strlen(s);
81         memcpy(result, s, (N + 1) * sizeof(char));
82         global_valid = 0;
83         int ret = get_possible_solution(s, result, len);
84         int colors[4] = {0};
85         get_broken_nums(result, s, colors, len);
86         printf("case #d:\n", i);
87         for (int j = 0; j < 3; j++) {
88             printf("%d ", colors[j]);
89         }
90         printf("%d\n", colors[3]);
91     }
92 }
93
94

```

./A.进制转换.c

```

1 #include<stdio.h>
2
3 char str[] = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";
4
5 int main(){
6     int t, n, r;
7     char result[33];
8     scanf("%d", &t);
9     for (int i=0; i<t; i++){
10        scanf("%d %d", &n, &r);
11        if (n < 0) {
12            n = - n;
13            printf("-");
14        }
15        int j = 0;
16        for (; n; n/=r, j++) result[j] = str[n%r];
17        while (--j >= 0) printf("%c", result[j]);
18        printf("\n");
19    }
20 }
21

```

./B.神秘信息.c

```

1 #include<stdio.h>
2
3 int main(){
4     int a[128];
5     int t, digit, n;
6     long long result;
7     char s[61], *p;
8     scanf("%d", &t);
9     for (int i=0; i<t; i++) {
10        printf("case #%d:\n", i);
11        for (int i=0; i<128; i++) a[i] = -1;
12        scanf("%s", s);
13        a[*s] = -1;
14        n = 1;
15        p = s;
16        digit = 0;
17        a[*p] = 1;
18        while (*++p){
19            if (a[*p] != -1) continue;
20            a[*p] = digit;
21            digit = digit ? digit + 1 : 2;
22            n ++;
23        }
24        if (n < 2) n = 2;
25        p = s;
26        result = 0;
27        while (*p)
28            result = result * n + a[*p++];
29        printf("%lld\n", result);

```



```
30     }
31 }
32
```

./C.按数据中1的位数排序.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  # define BIT_NUM 64
5  # define MAX_N 10000
6  typedef struct
7  {
8      long long value;
9      int number;
10 } Data;
11
12 int cmp(const void *pa, const void *pb){
13     Data a = *(Data*)pa;
14     Data b = *(Data*)pb;
15     if (a.number == b.number) {
16         if (a.value == b.value)
17             return 0;
18         else
19             return a.value > b.value ? 1 : -1;
20     } else
21         return b.number - a.number;
22 }
23
24 int main() {
25     int t, n;
26     Data nums[MAX_N];
27     scanf("%d", &t);
28     for (int i=0; i<t; i++) {
29         printf("case #%d:\n", i);
30         scanf("%d", &n);
31         for (int j=0; j<n; j++) {
32             scanf("%lld", &nums[j].value);
33             nums[j].number = 0;
34             long long d = 1;
35             for (int k=0; k<BIT_NUM; k++) {
36                 if (nums[j].value & d) nums[j].number++;
37                 d <<= 1;
38             }
39         }
40         qsort(nums, n, sizeof(nums[0]), cmp);
41         for (int j=0; j<n; j++)
42             printf("%lld ", nums[j].value);
43         printf("\n");
44     }
45 }
46
```

./D.内存显示.c

```
1 #include<stdio.h>
2 #include<string.h>
3 #include<stdlib.h>
4
5 void solveint(int n){
6     for (int i=0; i<sizeof(n); i++){
7         printf("%02x ", ((unsigned char*)&n)[i]);
8     }
9 }
10
11 void solvedouble(double d){
12     for (int i=0; i<sizeof(d); i++){
13         printf("%02x ", ((unsigned char*)&d)[i]);
14     }
15 }
16
17 int main(){
18     char s[31];
19     while (scanf("%s", s) != EOF){
20         if (strchr(s, '.') == 0)
21             solveint(atoi(s));
22         else
23             solvedouble(atof(s));
24         printf("\n");
25     }
26     return 0;
27 }
28
```

./E.平衡三进制.c

```
1 #include<stdio.h>
2
3 int main(){
4     int t;
5     char s[30];
6     scanf("%d", &t);
7     for (int i=0; i<t; i++){
8         printf("case #%d:\n", i);
9         scanf("%s", s);
10        int result = 0;
11        for (int j=0; s[j]; j++){
12            switch (s[j]){
13                case '-':
14                    result = result * 3 - 1;
15                    break;
16                case '0':
17                    result = result * 3;
18                    break;
19                case '1':
```

```

20         result = result * 3 + 1;
21         break;
22     }
23 }
24 printf("%d\n", result);
25 }
26 }
27

```

./F.非重复二进制串.c

```

1  #include<stdio.h>
2
3  int main(){
4      int t;
5      scanf("%d", &t);
6      for (int i=0; i<t; i++){
7          int n;
8          printf("case #d:\n", i);
9          scanf("%d", &n);
10         int max_diff_bin_len = 1, diff_bin_len = 1; // 初始值应为1, 因为如果大于
// 0的话, 一位肯定是不重复的。
11         for (; n > 1; n >>= 1){ // 最左的01不能算, 因为前置0不计算在内, 所以右移到01
// 时就可以停止了, 大于1代表可以继续右移。
12             switch (n & 0b11){
13                 case 0b10:
14                 case 0b01:
15                     diff_bin_len ++;
16                     if (diff_bin_len > max_diff_bin_len)
17                         max_diff_bin_len = diff_bin_len;
18                     break;
19                 case 0b00:
20                 case 0b11:
21                     diff_bin_len = 1;
22             }
23         }
24         printf("%d\n", max_diff_bin_len);
25     }
26 }
27

```

./G.二进制位不同的个数.c

```

1  #include<stdio.h>
2
3  int main(){
4      int t;
5      scanf("%d", &t);
6      for (int i=0; i<t; i++){
7          int x, y;
8          scanf("%d %d", &x, &y);
9          int distance = 0;

```

```

10     int xor = x ^ y;
11     for (; xor > 0; xor >>= 1){
12         distance += xor & 1;
13     }
14     printf("%d\n", distance);
15 }
16 }
17

```

./H.数据密度.c

```

1  #include<stdio.h>
2  #define MAX_LEN 130
3
4  int bin_with_1(char *pc){
5      unsigned char c = *(unsigned char*)pc; // 中文的每个字节首位可能是1
6      int result = 0;
7      do{
8          result += c & 1;
9      }while (c >>= 1);
10     return result;
11 }
12
13 /* 只能计算b > a情况时的最大公因数 */
14 int gcd(int a, int b){
15     if (a <= 0) return b;
16     int temp = b % a;
17     return gcd(temp, a);
18 }
19
20 int main(){
21     char s[500], *pchar;
22     int n;
23     scanf("%d", &n);
24     getchar(); // 因为scanf不会读取换行符, 所以要手动把第一个换行符读取了
25     while (n--){
26         fgets(s, MAX_LEN, stdin); // 会把结尾的换行符也读进来
27         // getline(&s, MAX_LEN, stdin);
28         int sum_with_bin_1 = 0;
29         for (pchar = s; (*pchar != '\n') && *pchar; pchar++){
30             sum_with_bin_1 += bin_with_1(pchar);
31         }
32         int sum_of_bin = (pchar - s) * 8;
33         int divisor = gcd(sum_with_bin_1, sum_of_bin);
34         printf("%d/%d\n", sum_with_bin_1 / divisor, sum_of_bin / divisor);
35     }
36 }
37

```

./I.QR Code.c

```

1  #include<stdio.h>

```

```

2  #include<stdlib.h>
3  #include<string.h>
4  #include<malloc.h>
5  #define NUM_SIGN "0001"
6  #define MAX_LEN 500
7  #define GROUP_LEN 3
8
9
10 void itob_print(int a, int bitlen){
11     char *result = (char*)malloc(bitlen + 1);
12     result[bitlen] = '\0';
13     while (--bitlen >= 0){
14         result[bitlen] = (a & 1) + '0';
15         a >>= 1;
16     }
17     printf(result);
18     free(result);
19 }
20
21 int main(){
22     char s[MAX_LEN + 1], *p_char;
23     scanf("%s", s);
24     printf(NUM_SIGN);
25     itob_print(strlen(s), 10);
26     int part_len=0, part_num=0;
27     for (p_char = s; *p_char; p_char++){
28         part_len ++;
29         part_num = part_num * 10 + *p_char - '0';
30         if (part_len >= GROUP_LEN){ // 每组的数字已完成
31             itob_print(part_num, 10);
32             part_len = 0;
33             part_num = 0;
34         }
35     }
36     switch (part_len){ // 还有剩余的一些数字没输出
37         case 1:
38             itob_print(part_num, 4);
39             break;
40         case 2:
41             itob_print(part_num, 7);
42             break;
43     }
44 }
45 }
46

```

./J.i-1进制 (Easy) .c

```

1  // 实部
2  // sqrt s
3  // 0 s2 2 2s2 4 4s2 8 8s2 16 16s2
4  // 虚部
5  // 0 3 6 9 12 15 18 21 24 27 30
6  // 0 3 6 1 4 7 2 5 0 3 6

```

```

7 // 18:04
8 #include<stdio.h>
9 #include<string.h>
10
11 typedef struct {
12     long long real;
13     long long imaginary;
14 } Complex;
15
16 /* 计算-1 + i的幂 */
17 Complex pow_of_complex(int multiple_times) {
18     // if (multiple_times <= 0) return {1LL, 0LL};
19     long long modulus = 1LL << (multiple_times / 2); // 偶数时是模长的意思, 奇数
    时是实部或虚部的绝对值 (实部和虚部绝对值相等)
20     Complex result = {0LL, 0LL};
21     switch (multiple_times % 8) {
22         case 0: // 0 * 45°
23             result.real = modulus;
24             break;
25         case 1: // 3 * 45°
26             result.real = - modulus;
27             result.imaginary = modulus;
28             break;
29         case 2: // 6 * 45°
30             result.imaginary = - modulus;
31             break;
32         case 3: // 1 * 45°
33             result.real = modulus;
34             result.imaginary = modulus;
35             break;
36         case 4: // 4 * 45°
37             result.real = - modulus;
38             break;
39         case 5: // 7 * 45°
40             result.real = modulus;
41             result.imaginary = - modulus;
42             break;
43         case 6: // 2 * 45°
44             result.imaginary = modulus;
45             break;
46         case 7: // 5 * 45°
47             result.real = - modulus;
48             result.imaginary = - modulus;
49             break;
50     }
51     return result;
52 }
53
54 /* 两个复数即使在不需要二进制进位的情况下相加, 也不能直接使用异或, 因为负数的时候异或就不对了 */
55 Complex add(Complex a, Complex b) {
56     Complex result;
57     result.real = a.real + b.real;
58     result.imaginary = a.imaginary + b.imaginary;
59     return result;

```

```

60 }
61
62 int main(){
63     char str_hex_num[100];
64     Complex result = {0LL, 0LL};
65     scanf("0x%s", str_hex_num);
66     char *pos_hex_num = str_hex_num;
67     for (int weight=strlen(str_hex_num) - 1; weight>=0; weight--,
pos_hex_num++){
68         int hex_num;
69         if (*pos_hex_num <= '9')
70             hex_num = *pos_hex_num - '0';
71         else
72             hex_num = *pos_hex_num - 'A' + 10;
73         for (int i=0; hex_num>0; hex_num >>= 1, i++){
74             if (hex_num & 1)
75                 result = add(result, pow_of_complex(4 * weight + i));
76         }
77     }
78     if (result.real > 0 && 4 * strlen(str_hex_num) >= 128 && (*str_hex_num
== '1' || *str_hex_num == 'E'))
79         result.real --; // 看了测试用例后发现超出范围的都是输出比答案多了1, 所以手动判
断是否超出范围
80     if (result.real != 0 && result.imaginary > 0 && 4 * strlen(str_hex_num)
>= 128 && (*str_hex_num == '3' || *str_hex_num == '7' || *str_hex_num ==
'E'))
81         result.imaginary --; // 看了测试用例后发现超出范围的都是输出比答案多了1, 所以
手动判断是否超出范围
82     if (result.imaginary < 0 && 4 * strlen(str_hex_num) >= 128 &&
(*str_hex_num == '7' ))
83         result.imaginary ++; // 看了测试用例后发现超出范围的都是输出比答案多了1, 所以
手动判断是否超出范围
84     if (result.imaginary == 0) {
85         printf("%lld", result.real);
86     }else if (result.real == 0 ) {
87         if (result.imaginary == 1) printf("i");
88         else if (result.imaginary == -1) printf("-i");
89         else printf("%lldi", result.imaginary);
90     }else{
91         if (result.imaginary == 1) printf("%lld+i", result.real);
92         else if (result.imaginary == -1) printf("%lld-i", result.real);
93         else printf("%lld%lldi", result.real, result.imaginary);
94     }
95     return 0;
96 }
97 // 20:31
98

```

./K.-1+i进制.c

```

1 // 21:13
2 #include<stdio.h>
3 #include<regex.h>
4 #include<string.h>

```

```

5 #include<stdlib.h>
6 #include<malloc.h>
7 #define MAX_E 50 // 最大位数的整数
8 #define MAX_LINE 100 // 输入的一行最大字符
9
10 typedef struct {
11     long long real;
12     long long imaginary;
13 } Complex;
14
15 regex_t reg_full, reg_real, reg_imaginary;
16 regmatch_t matches[10];
17 int ret = 0;
18 int cflags = REG_EXTENDED | REG_ICASE | REG_NEWLINE;
19 char input[MAX_LINE] = {'\0'};
20
21 int getsign(char *input, regmatch_t *match) {
22     // printf("%s %d %d\n", input, match->rm_so, match->rm_eo);
23     if (match->rm_so < 0 || match->rm_so == match->rm_eo) // 省略正号
24         return 1;
25     else if (*(input + match->rm_so) == '+')
26         return 1;
27     return -1;
28 }
29
30 long long getabs(char *input, regmatch_t *match) {
31     // printf("%s %d %d\n", input, match->rm_so, match->rm_eo);
32     if (match->rm_so < 0 || match->rm_so == match->rm_eo) // 省略1
33         return 1;
34     char *abs_str = malloc((match->rm_eo - match->rm_so + 1) *
sizeof(char));
35     int i;
36     for (i = match->rm_so; i < match->rm_eo; i++) {
37         abs_str[i - match->rm_so] = input[i];
38     }
39     abs_str[i - match->rm_so] = '\0'; // 之前这里少了 - match->rm_so
40     // char *abs_str = strdup(input + match->rm_so, match->rm_eo - match-
->rm_so); // 使用这个在本地运行不会报错, 在OJ运行会Runtime error: SIGSEGV
41     long long result = atoll(abs_str);
42     free(abs_str);
43     return result;
44 }
45
46 Complex get_complex(char *input, int limit) {
47     Complex input_complex = {0LL, 0LL};
48     fgets(input, limit, stdin);
49     ret = regexec(&reg_full, input, 5, matches, 0);
50     if (ret) { // 省略了实部或虚部
51         if (strchr(input, 'i') == NULL) { // 只有实部
52             ret = regexec(&reg_real, input, 3, matches, 0);
53             input_complex.real = getsign(input, matches + 1)
* getabs(input, matches + 2);
54         } else { // 只有虚部
55             ret = regexec(&reg_imaginary, input, 3, matches, 0);
56             input_complex.imaginary = getsign(input, matches + 1)

```



```

58         * getabs(input, matches + 2);
59     }
60     } else {
61         input_complex.real = getsign(input, matches + 1)
62             * getabs(input, matches + 2);
63         input_complex.imaginary = getsign(input, matches + 3)
64             * getabs(input, matches + 4);
65     }
66     return input_complex;
67 }
68
69 Complex divide_once(Complex a) {
70     Complex result;
71     // >>1是向下取整, /2是向0取整
72     result.real = a.imaginary / 2 - a.real / 2;
73     result.imaginary = - a.real / 2 - a.imaginary / 2;
74     if (a.real & 1) { // 这里已经确保实部虚部奇偶性相同, 实部虚部都为奇数时上面两行会
75         // 偏差1, 对负数用%2会出现问题, 所以应用&1来判断奇偶
76         if (a.real < 0 && a.imaginary < 0)
77             result.imaginary ++; // 因为向0取整, 所以向远离0的地方偏移1
78         if (a.real > 0 && a.imaginary > 0)
79             result.imaginary --;
80         if (a.real < 0 && a.imaginary > 0)
81             result.real ++; // 异号时实部会有偏差
82         if (a.real > 0 && a.imaginary < 0)
83             result.real --;
84     }
85     return result;
86 }
87
88 void print_bin_complex(Complex a) {
89     if (a.imaginary == 0 && (a.real == 1 || a.real == 0)){
90         printf("%lld", a.real);
91         return;
92     } else if ((a.real & 1) != (a.imaginary & 1)) { // 负奇数对2取余是-1, 因此
93         // 要取绝对值, 但用&1来判断就不用取绝对值了
94         a.real --;
95         print_bin_complex(divide_once(a));
96         printf("1");
97     } else {
98         print_bin_complex(divide_once(a));
99         printf("0");
100     }
101 }
102
103 // 21:42
104 // 11:00
105 // 11:51
106 // 13:49
107 // 17:03
108 // 19:20
109 // 20:38
110
111 int main(){
112     // Complex input_complex = {"0", POS}, {"0", POS}};

```

```

110     regcomp(&reg_full, "(-)?([0-9]+)(\\+|-)([0-9]*)i$", cflags); // 加入
REG_NEWLINE使^$忽略换行符, 不加^来忽略可能不省略的分号
111     regcomp(&reg_real, "(-)?([0-9]+)$", cflags); // 加入REG_NEWLINE使^$忽略换
行符
112     regcomp(&reg_imaginary, "(-)?([0-9]*)i$", cflags); // 加入REG_NEWLINE使
^$忽略换行符
113     Complex input_complex = get_complex(input, MAX_LINE - 1);
114     // printf("%lld %lld\n", input_complex.real, input_complex.imaginary);
115     // Complex temp_complex = get_complex(input, MAX_LINE);
116     print_bin_complex(input_complex);
117 }
118

```

./K.-1+i进制 (长整数加减法未完成) .c

```

1 // 21:13
2 #include<stdio.h>
3 #include <sys/types.h>
4 #include <regex.h>
5 #include<string.h>
6 #define MAX_E 50 // 最大位数的整数
7 #define MAX_LINE 100 // 输入的一行最大字符
8
9 typedef struct {
10     char *abs; // char abs[MAX_E]
11     enum {NEG=-1, POS=1} sign;
12 } Decimal;
13
14 typedef struct {
15     Decimal real;
16     Decimal imaginary;
17 } Complex;
18
19 regex_t reg_full, reg_real, reg_imaginary;
20
21 Decimal add(Decimal a, Decimal b) {
22     char abs[MAX_E] = "0";
23     Decimal result={abs, POS};
24     if (a.sign == b.sign) {
25         result.sign = a.sign;
26         for (int i=MAX_E; i>0; i--) {
27             int temp = a.abs[i] + b.abs[i];
28             result.abs[i] = temp % 10;
29             result.abs[i - 1] = temp / 10;
30         }
31     } else if (a.abs > b.abs) {
32         result.sign = a.sign;
33         for (int i=MAX_E; i>0; i--) {
34             int temp = a.abs[i] - b.abs[i];
35             result.abs[i] += temp % 10;
36             result.abs[i - 1] += temp / 10;
37         }
38     } else if (a.abs < b.abs) {
39         result.sign = b.sign;

```

```

40     for (int i=MAX_E; i>0; i--) {
41         int temp = b.abs[i] - a.abs[i];
42         result.abs[i] += temp % 10;
43         result.abs[i - 1] += temp / 10;
44     }
45 }
46 return result;
47 }
48
49 int getsign(char *input, regmatch_t *match) {
50     if (match->rm_eo == match->rm_so)
51         return POS;
52     else if (*(input + match->rm_so) == '+')
53         return POS;
54     return NEG;
55 }
56
57 char *getabs(char *input, regmatch_t *match) {
58     if (match->rm_eo == match->rm_so) // 省略1
59         return "1";
60     return strdup(input + match->rm_so, match->rm_eo - match->rm_so);
61 }
62
63 Complex get_complex(char *input, int limit) {
64     int ret;
65     regmatch_t matches[5];
66     Complex input_complex = {"0", POS}, {"0", POS}};
67     fgets(input, MAX_LINE, stdin);
68     ret = regexec(&reg_full, input, 5, matches, 0);
69     if (ret) { // 省略了实部或虚部
70         if (strchr(input, 'i') == NULL) { // 只有实部
71             ret = regexec(&reg_real, input, 3, matches, 0);
72             input_complex.real.sign = getsign(input, matches + 1);
73             input_complex.real.abs = getabs(input, matches + 2);
74         } else { // 只有虚部
75             ret = regexec(&reg_imaginary, input, 3, matches, 0);
76             input_complex.imaginary.sign = getsign(input, matches + 1);
77             input_complex.imaginary.abs = getabs(input, matches + 2);
78         }
79     } else {
80         input_complex.real.sign = getsign(input, matches + 1);
81         input_complex.real.abs = getabs(input, matches + 2);
82         input_complex.imaginary.sign = getsign(input, matches + 3);
83         input_complex.imaginary.abs = getabs(input, matches + 4);
84     }
85     return input_complex;
86 }
87
88 // 21:42
89 // 11:00
90 // 11:51
91 // 13:49
92 int main(){
93     int ret;
94     int cflags = REG_EXTENDED | REG_ICASE | REG_NEWLINE;

```

```

95     char input[MAX_LINE];
96     // Complex input_complex = {"0", POS}, {"0", POS}};
97     regcomp(&reg_full, "(-)?([0-9]+)(\\+|-)([0-9]*)i$", cflags); // 加入
REG_NEWLINE使^$忽略换行符, 不加^来忽略可能不省略的分号
98     regcomp(&reg_real, "(-)?([0-9]+)$", cflags); // 加入REG_NEWLINE使^$忽略换
行符
99     regcomp(&reg_imaginary, "(-)?([0-9]*)i$", cflags); // 加入REG_NEWLINE使
^$忽略换行符
100    Complex input_complex = get_complex(input, MAX_LINE);
101    printf("%d %s %d %s\n",
102           input_complex.real.sign, input_complex.real.abs,
103           input_complex.imaginary.sign, input_complex.imaginary.abs);
104    // Complex temp_complex = get_complex(input, MAX_LINE);
105    printf("%d %s\n",
106           add(input_complex.real, input_complex.imaginary).sign,
107           add(input_complex.imaginary, input_complex.real).abs);
108 }
109

```

./L.平衡三进制.c

```

1 // 9:37
2 // >>> import math
3 // >>> math.log(3**30, 10)
4 // 14.313637641589873
5
6 #include<stdio.h>
7 #include<regex.h>
8 #include<stdlib.h>
9 #define MAX_LEN 50
10 #define BASE 3
11
12 /* 计算a < b时a和b的最大公因数 */
13 long long gcd(long long a, long long b) {
14     if (a == 0) return b;
15     return gcd(b % a, a);
16 }
17
18 long long tri_to_ll(char *a, int len) {
19     long long result = 0;
20     for (char *pos_num = a; pos_num < a + len; pos_num++) {
21         int num = *pos_num - '0';
22         if (num == 2) num = -1;
23         result = result * BASE + num;
24     }
25     return result;
26 }
27
28 int main() {
29     char tri[MAX_LEN];
30     fgets(tri, MAX_LEN - 1, stdin);
31     regex_t reg;
32     regmatch_t matches[3];
33     int cflags = REG_NEWLINE | REG_EXTENDED | REG_ICASE;

```

```

34     regcomp(&reg, "[0-9+]\.\.?([0-9]*)", cflags);
35     int ret = regexec(&reg, tri, 3, matches, 0);
36     // for (int i=1; i<3; i++) {
37     //     if (matches[i].rm_so < 0) continue;
38     //     for (int j=matches[i].rm_so; j<matches[i].rm_eo; j++){
39     //         printf("%c", tri[j]);
40     //     }
41     //     printf("\n");
42     // }
43     long long int_part = tri_to_ll(tri + matches[1].rm_so, matches[1].rm_eo
- matches[1].rm_so);
44     long long numerator = tri_to_ll(tri + matches[2].rm_so, matches[2].rm_eo
- matches[2].rm_so);
45     long long denominator = 1;
46     for (int point_pos = matches[2].rm_eo - matches[2].rm_so; point_pos > 0;
point_pos--) {
47         denominator *= 3;
48     }
49     if (int_part > 0 && numerator < 0) { // 借位
50         int_part --;
51         numerator += denominator;
52     }
53     if (int_part < 0 && numerator > 0) { // 借位
54         int_part ++;
55         numerator = denominator - numerator;
56     }
57     long long factor = gcd(llabs(numerator), llabs(denominator));
58     if (int_part || numerator==0)
59         printf("%lld ", int_part);
60     if (numerator) {
61         printf("%lld %lld", numerator / factor, denominator / factor);
62     }
63 }
64 // 10:52
65

```

./M.平衡三进制II.c

```

1 // 10:53
2 // >>> math.log(10**9,3)
3 // 18.86312946860446
4 #include<stdio.h>
5 #include<stdlib.h>
6 #include<string.h>
7 #define MAX_LEN 50
8
9 int get_point_pos(long long denominator) {
10     int point_pos = 0;
11     while (denominator > 1) {
12         denominator /= 3;
13         point_pos ++;
14     }
15     return point_pos;
16 }

```

```

17
18 /* 十进制转三进制, 返回三进制的字符个数 */
19 int dec_to_tri(long long dec, char *tri) {
20     if (dec < 3) {
21         *tri = dec + '0';
22         return 1;
23     }
24     int len = dec_to_tri(dec / 3, tri);
25     *(tri + len) = dec % 3 + '0';
26     return len + 1;
27 }
28
29 /* 注意, 这里的target-1必须是一个合法的地址, 用来存放进位 */
30 int add_tri(char *target, char *addition, int len) {
31     int carry = 0; // 存储进位
32     for (int i = len - 1; i >= 0; i--) {
33         target[i] += addition[i] - '0' + carry;
34         carry = 0;
35         if (target[i] >= '3') {
36             carry = 1;
37             target[i] -= 3;
38         }
39     }
40     if (carry) // 最后一次进位
41         target[-1] ++;
42 }
43
44 int sub_ones(char *target, int len) {
45     for (char *i = target; i < target + len; i++) {
46         (*i) --;
47         if (*i < '0') *i = '2';
48     }
49 }
50
51 void my_strncpy(char *dest, char *src, int limit) {
52     char *p_dest = dest, *p_src = src;
53     while (*p_src && limit > 0) {
54         *p_dest++ = *p_src++;
55         limit --;
56     }
57     *p_dest = 0;
58 }
59
60 char *simplify_int_part(char *part, int len) {
61     while (len > 1 && *part == '0')
62         part++, len--;
63     return part;
64 }
65
66 int simplify_fraction_part(char *part, int len) {
67     while (len > 0 && part[--len] == '0')
68         part[len] = 0;
69     if (part[len] == '0' || part[len] == 0) return len;
70     else return len + 1;
71 }

```

```

72
73 // 11:43
74 // 12:48
75 // 14:43
76
77 int main() {
78     long long numerator, denominator;
79     scanf("%lld %lld", &numerator, &denominator);
80     char tri[MAX_LEN], ones[MAX_LEN];
81     tri[0] = '0';
82     int len = dec_to_tri(numerator, tri + 1); // 留出第一位可能的进位空间
83     tri[1 + len] = '\0';
84     // printf("%s\n", tri);
85     for (int i = 0; i < len; i++)
86         ones[i] = '1'; // 构造全1序列
87     ones[len] = '\0';
88     add_tri(tri + 1, ones, len);
89     // printf("%s\n", tri);
90     sub_ones(tri + 1, len);
91     // printf("%s\n", tri);
92     int point_pos = get_point_pos(denominator);
93     // char *p_endpos = tri + 1 + len;
94     // char *output = tri;
95     // while (*output == '0') output++; // 去除前面的0
96     // if (output >= p_endpos - point_pos) putchar('0'); // 纯小数
97     // while (output < p_endpos - point_pos) putchar(*output++); // 输出整数
    部分
98     // while (*(p_endpos - 1) == '0') p_endpos--; // 去除后面的0
99     // if (point_pos > 0 && output < p_endpos) { // 判断是否需要输出小数部分
100     //     putchar('.');
101     //     for (int i = point_pos - (p_endpos - output); i > 1; i--) {
102     //         putchar('0'); // 纯小数
103     //     }
104     //     while (output < p_endpos) putchar(*output++);
105     // }
106     int full_len = len + 1; // 三进制数字总长度
107     char tri_with_start_0[MAX_LEN];
108     int i = 0;
109     for (; i < point_pos - full_len + 1; i++) {
110         tri_with_start_0[i] = '0';
111     }
112     int j = 0;
113     while (tri[j]) tri_with_start_0[i++] = tri[j++];
114     tri_with_start_0[i] = '\0';
115     full_len = strlen(tri_with_start_0);
116     char int_part[MAX_LEN], fraction_part[MAX_LEN];
117     my_strncpy(int_part, tri_with_start_0, full_len - point_pos);
118     my_strncpy(fraction_part, tri_with_start_0 + full_len - point_pos,
point_pos);
119     char *new_int_part = simplify_int_part(int_part, strlen(int_part));
120     int fraction_len = simplify_fraction_part(fraction_part,
strlen(fraction_part));
121     printf("%s", new_int_part);
122     if (fraction_len > 0)
123         printf(".%s", fraction_part);

```

```

124     return 0;
125 }
126
127

```

./library/大整数计算1.c

```

1 // https://blog.csdn.net/u013113678/article/details/113060506
2
3 #include<stdint.h>
4 #include<stdio.h>
5 #include<string.h>
6 //计算位数, 范围[1,9]
7 #define NUM_DIGIT 9
8 //数组类型
9 #if NUM_DIGIT<=0
10     static_assert(1, "NUM_DIGIT must > 0 and <=9");
11 #elif NUM_DIGIT <=2
12 #define _NUM_T int8_t
13 #elif NUM_DIGIT <=4
14 #define _NUM_T int16_t
15 #elif NUM_DIGIT <=9
16 //NUM_DIGIT在[5,9]时用int32就可以装载数据, 但是实测发现用int32在32位程序中, 对除法性
    能有影响, 用int64反而正常, 这是比较奇怪的现象。
17 #define _NUM_T int64_t
18 #else
19     static_assert(1, "NUM_DIGIT must > 0 and <=9");
20 #endif
21 //计算进制
22 static const size_t _hex = NUM_DIGIT == 1 ? 10 : NUM_DIGIT == 2 ? 100 :
    NUM_DIGIT == 3 ? 1000 : NUM_DIGIT == 4 ? 10000 : NUM_DIGIT == 5 ? 100000 :
    NUM_DIGIT == 6 ? 1000000 : NUM_DIGIT == 7 ? 10000000 : NUM_DIGIT == 8 ?
    100000000 : NUM_DIGIT == 9 ? 1000000000 : -1;
23 #define _MIN_NUM_SIZE 30/NUM_DIGIT
24
25     /// <summary>
26     /// 通过字符串初始化
27     /// </summary>
28     /// <param name="a">[in]高精度数组</param>
29     /// <param name="value">[in]字符串首地址</param>
30     /// <param name="len">[in]字符串长度</param>
31     /// <returns>数据长度</returns>
32 static size_t initByStr(_NUM_T* a, const char* value, size_t len)
33 {
34     size_t shift = 10;
35     size_t i, j, k = 0, n;
36     n = len / NUM_DIGIT;
37     for (i = 0; i < n; i++)
38     {
39         a[i] = (value[len - k - 1] - '0');
40         for (j = 1; j < NUM_DIGIT; j++)
41         {
42             a[i] += (value[len - k - j - 1] - '0') * shift;
43             shift *= 10;

```



```

44     }
45     shift = 10;
46     k += NUM_DIGIT;
47 }
48 if (n = len - n * NUM_DIGIT)
49 {
50     a[i] = (value[len - k - 1] - '0');
51     for (j = 1; j < n; j++)
52     {
53         a[i] += (value[len - k - j - 1] - '0') * shift;
54         shift *= 10;
55     }
56     i++;
57 }
58 return i;
59 }
60 /// <summary>
61 /// 通过无符号32整型初始化
62 /// </summary>
63 /// <param name="a">[in]高精度数组</param>
64 /// <param name="value">[in]整型值</param>
65 /// <returns>数据长度</returns>
66 static size_t initByInt32(_NUM_T* a, uint32_t value)
67 {
68     size_t i;
69     for (i = 0; i < 8096; i++)
70     {
71         a[i] = value % _hex;
72         value /= _hex;
73         if (!value)
74         {
75             i++;
76             break;
77         }
78     }
79     return i;
80 }
81 /// <summary>
82 /// 通过无符号64整型初始化
83 /// </summary>
84 /// <param name="a">[in]高精度数组</param>
85 /// <param name="value">[in]整型值</param>
86 /// <returns>数据长度</returns>
87 static size_t initByInt64(_NUM_T* a, uint64_t value)
88 {
89     size_t i;
90     for (i = 0; i < 8096; i++)
91     {
92         a[i] = value % _hex;
93         value /= _hex;
94         if (!value)
95         {
96             i++;
97             break;
98         }

```

```

99     }
100     return i;
101 }
102 /// <summary>
103 /// 比较两个高精度数的大小
104 /// </summary>
105 /// <param name="a">[in]第一个数</param>
106 /// <param name="aLen">[in]第一个数的长度</param>
107 /// <param name="b">[in]第二个数</param>
108 /// <param name="bLen">[in]第二个数的长度</param>
109 /// <returns>1是a>b,0是a==b,-1是a<b</returns>
110 static int compare(const _NUM_T* a, size_t aLen, const _NUM_T* b,
size_t bLen)
111 {
112     if (aLen > bLen)
113     {
114         return 1;
115     }
116     else if (aLen < bLen)
117     {
118         return -1;
119     }
120     else {
121         for (int i = aLen - 1; i > -1; i--)
122         {
123             if (a[i] > b[i])
124             {
125                 return 1;
126             }
127             else if (a[i] < b[i])
128             {
129                 return -1;
130             }
131         }
132     }
133     return 0;
134 }
135 /// <summary>
136 /// 打印输出结果
137 /// </summary>
138 static void print(const _NUM_T* a, size_t aLen) {
139     int i = aLen - 1, j = 0, n = _hex;
140     char format[32];
141     sprintf(format, "%0%dd", NUM_DIGIT);
142     printf("%d", a[i--]);
143     for (; i > -1; i--)
144         printf(format, a[i]);
145 }
146
147 /// <summary>
148 /// 加法(累加)
149 ///结果会保存在a中
150 /// </summary>
151 /// <param name="a">[in]被加数</param>
152 /// <param name="aLen">[in]被加数长度</param>

```

```

153     /// <param name="b">[in]加数</param>
154     /// <param name="bLen">[in]加数长度</param>
155     /// <returns>结果长度</returns>
156     static size_t acc(_NUM_T* a, size_t aLen, const _NUM_T* b, size_t
bLen)
157     {
158         size_t i, n = bLen;
159         const size_t max = _hex - 1;
160         if (aLen < bLen)
161         {
162             memset(a + aLen, 0, (bLen - aLen + 1) * sizeof(_NUM_T));
163         }
164         else {
165             a[aLen] = 0;
166         }
167         for (i = 0; i < bLen; i++) {
168             uint64_t temp = (uint64_t)a[i] + (uint64_t)b[i];
169             a[i] = temp % _hex;
170             a[i + 1] += temp / _hex;
171         }
172         for (; i < aLen; i++) {
173             uint64_t temp = a[i];
174             if (temp <= max)
175                 break;
176             a[i] = temp % _hex;
177             a[i + 1] += temp / _hex;
178         }
179         n = aLen > bLen ? aLen : bLen;
180         if (a[n])
181             return n + 1;
182         return n;
183     }
184
185     /// <summary>
186     /// 减法(累减)
187     ///结果会保存在a中
188     /// </summary>
189     /// <param name="a">[in]被减数, 被减数必须大于等于减数</param>
190     /// <param name="aLen">[in]被减数长度</param>
191     /// <param name="b">[in]减数</param>
192     /// <param name="bLen">[in]减数长度</param>
193     /// <returns>结果长度</returns>
194     static size_t subc(_NUM_T* a, size_t aLen, const _NUM_T* b, size_t
bLen) {
195         size_t m, n, i;
196         if (aLen < bLen)
197         {
198             m = bLen;
199             n = aLen;
200         }
201         else {
202             n = bLen;
203             m = aLen;
204         }
205         for (i = 0; i < n; i++)

```

```

206     {
207         int64_t temp = (int64_t)a[i] - (int64_t)b[i];
208         a[i] = temp;
209         if (temp < 0)
210         {
211             a[i + 1] -= 1;
212             a[i] += _hex;
213         }
214     }
215     for (; i < m; i++)
216     {
217         _NUM_T temp = a[i];
218         if (temp < 0)
219         {
220             a[i + 1] -= 1;
221             a[i] += _hex;
222         }
223         else
224         {
225             break;
226         }
227     }
228     for (size_t i = aLen - 1; i != SIZE_MAX; i--)
229         if (a[i])
230             return i + 1;
231     return 1;
232 }
233
234 /// <summary>
235     /// 乘法
236     /// </summary>
237     /// <param name="a">[in]被乘数</param>
238     /// <param name="aLen">[in]被乘数长度</param>
239     /// <param name="b">[in]乘数</param>
240     /// <param name="bLen">[in]乘数长度</param>
241     /// <param name="c">[out]结果，数组长度必须大于等于aLen+bLen+1，且全部元素为
242     0</param>
243     /// <returns>结果长度</returns>
244     static size_t multi(const _NUM_T* a, size_t aLen, const _NUM_T* b,
245     size_t bLen, _NUM_T c[]) {
246         _NUM_T d;
247         size_t j;
248         c[aLen + bLen] = 0;
249         for (size_t i = 0; i < aLen; i++)
250         {
251             d = 0;
252             for (j = 0; j < bLen; j++)
253             {
254                 uint64_t temp = (uint64_t)a[i] * (uint64_t)b[j] + c[j + i]
255                 + d;
256                 d = temp / _hex;
257                 c[j + i] = temp % _hex;
258             }
259             if (d)
260             {

```

```

258         c[j + i] = d;
259     }
260 }
261 for (size_t k = aLen + bLen; k != SIZE_MAX; k--)
262     if (c[k])
263         return k + 1;
264 return 1;
265 }
266
267 /// <summary>
268     /// 除法
269     /// </summary>
270     /// <param name="a">[in]被除数,被除数必须大于除数。小于商为0、余数为除数,等于商
    为1、余数为0,不需要在函数内实现,在外部通过compare就可以做到。</param>
271     /// <param name="aLen">[in]被除数长度</param>
272     /// <param name="b">[in]除数</param>
273     /// <param name="bLen">[in]除数长度</param>
274     /// <param name="c">[out]商,数组长度大于等于aLen,且全部元素为0</param>
275     /// <param name="mod">[out]余数,数组长度大于等于aLen</param>
276     /// <param name="modLen">[out]余数长度</param>
277     /// <param name="temp">[in]临时缓冲区,由外部提供以提高性能,数组长度大于等于
    aLen+bLen+1</param>
278     /// <returns>商长度</returns>
279     static size_t divide(const _NUM_T* a, size_t aLen, const _NUM_T*
    b, size_t bLen, _NUM_T* c, _NUM_T* mod, size_t* modLen, _NUM_T* temp) {
280         intptr_t n, l;
281         int equal;
282         memcpy(mod, a, (aLen) * sizeof(_NUM_T));
283         n = aLen - bLen;
284         l = aLen;
285         while (n > -1)
286         {
287             equal = compare(mod + n, l - n, b, bLen);
288             if (equal == -1)
289             {
290                 n--;
291                 if (!mod[l - 1])
292                     l--;
293                 continue;
294             }
295             if (equal == 0)
296             {
297                 c[n]++;
298                 n -= bLen;
299                 l -= bLen;
300                 continue;
301             }
302             uint64_t x;
303             if ((l - n) > bLen)
304             {
305                 x = ((mod[l - 1]) * _hex) / ((uint64_t)b[bLen - 1] + 1);
306                 if (x == 0)
307                 {
308                     x = (mod[l - 2]) / ((uint64_t)b[bLen - 1] + 1);
309                 }

```

```

310     }
311     else
312     {
313         x = (mod[l - 1]) / ((uint64_t)b[bLen - 1] + 1);
314     }
315     if (x == 0)
316         x = 1;
317     c[n] += x;
318     if (x == 1)
319     {
320         l = n + subc(mod + n, l - n, b, bLen);
321     }
322     else
323     {
324         _NUM_T num[_MIN_NUM_SIZE];
325         size_t len = initByInt32(num, x);
326         memset(temp, 0, (aLen + bLen + 1) * sizeof(_NUM_T));
327         len = multi(b, bLen, num, len, temp);
328         l = n + subc(mod + n, l - n, temp, len);
329     }
330 }
331 intptr_t i = l - 1;
332 for (; i > -1; i--)
333     if (mod[i])
334         break;
335 if (i == -1)
336 {
337     mod[0] = 0;
338     *modLen = 1;
339 }
340 else
341 {
342     *modLen = i + 1;
343 }
344 if (c[aLen - bLen] != 0)
345     return aLen - bLen + 1;
346 else
347     return aLen - bLen;
348 }
349
350 int sum(){
351     int64_t n;
352     size_t len, len2;
353     _NUM_T num[1024];
354     _NUM_T num2[1024];
355     scanf("%ld", &n);
356     len = initByInt32(num, 0);
357     for (int64_t i = 1; i <= n; i++)
358     {
359         len2 = initByInt64(num2, i);
360         len = acc(num, len, num2, len2);
361     }
362     print(num, len);
363     return 0;
364 }

```

```

365
366 int minus()
367 {
368     _NUM_T a1[8096], a2[8096];
369     char s1[8096], s2[8096];
370     scanf("%s%s", s1, s2);
371     size_t len1, len2;
372     len1 = initByStr(a1, s1, strlen(s1));
373     len2 = initByStr(a2, s2, strlen(s2));
374     len1 = subc(a1, len1, a2, len2);
375     print(a1, len1);
376     return 0;
377 }
378
379 int factorial(){
380     int64_t n;
381     size_t len, len2;
382     _NUM_T num[8096];
383     _NUM_T num2[8096];
384     _NUM_T num3[8096];
385     _NUM_T* p1 = num;
386     _NUM_T* p2 = num3;
387     scanf("%ld", &n);
388     len = initByInt32(num, 1);
389     for (int64_t i = 1; i <= n; i++)
390     {
391         len2 = initByInt64(num2, i);
392         memset(p2, 0, 8096 * sizeof(_NUM_T));
393         len = multi(p1, len, num2, len2, p2);
394         _NUM_T* temp = p1;
395         p1 = p2;
396         p2 = temp;
397     }
398     print(p1, len);
399     return 0;
400 }
401
402 int division()
403 {
404     _NUM_T a1[8096], a2[8096], c[8096], mod[8096], temp[8096];
405     char s1[8096], s2[8096];
406     scanf("%s%s", s1, s2);
407     size_t len1, len2, ml;
408     len1 = initByStr(a1, s1, strlen(s1));
409     len2 = initByStr(a2, s2, strlen(s2));
410     memset(c, 0, 8096 * sizeof(_NUM_T));
411     len1 = divide(a1, len1, a2, len2, c, mod, &ml, temp);
412     print(c, len1);
413     printf("\n");
414     print(mod, ml);
415     return 0;
416 }
417
418 int main() {

```

```

419     printf("求 n 累加和(ja)。用高精度方法, 求 s=1+2+3+.....+n 的精确值(n 以一般整数输
      入)。 \n");
420     sum();
421     printf("\n两个任意十一位数的减法; (小于二十位) \n");
422     minus();
423     printf("\n求 N! 的值(ni)。用高精度方法, 求 N! 的精确值(N 以一般整数输入)。 \n");
424     factorial();
425     printf("\n给定两个非负整数A, B, 请你计算 A / B的商和余数。 \n");
426     division();
427 }
428

```

./library/大整数计算2.c

```

1 // https://www.cnblogs.com/HappyXie/articles/1927557.html
2
3 #include <stdio.h>
4 #include <stdlib.h>
5 #define MAX 200
6
7 typedef struct
8 {
9     int len;
10    int s[MAX+1];
11 } hp;
12
13 void input(hp *a, int x) //读入数字
14 {
15     int i;
16
17     a->len = 0;
18
19     while (x > 0)
20     {
21         a->s[1 + a->len++] = x % 10;
22         x /= 10;
23     }
24
25     for (i = a->len + 1; i <= MAX; i++)
26         a->s[i] = 0;
27 }
28
29 void input1(hp *a, char *str) //读入字符串
30 {
31     int i, len;
32
33     a->len = 0;
34
35     if (str == NULL)
36         return;
37
38     len = strlen(str);
39
40     while (len > 0)

```



```

41     {
42         a->s[1 + a->len++] = *(str + len - 1) - '0';
43         len--;
44     }
45
46     for (i = a->len + 1; i <= MAX; i++)
47         a->s[i] = 0;
48 }
49
50 void print(hp *y) //打印数字
51 {
52     int i;
53     for (i = y->len; i >= 1; i--)
54         printf("%d", y->s[i]);
55     printf("\n");
56 }
57
58 void add(hp *a, hp *b, hp *c) //高精度加法 c = a + b
59 {
60     int i, len;
61
62     for (i = 1; i <= MAX; i++) c->s[i] = 0;
63
64     if (a->len > b->len) len = a->len;
65     else len = b->len;
66
67     for (i = 1; i <= len; i++)
68     {
69         c->s[i] += a->s[i] + b->s[i];
70         if (c->s[i] >= 10)
71         {
72             c->s[i] -= 10;
73             c->s[i+1]++;
74         }
75     }
76
77     if (c->s[len+1] > 0) len++;
78     c->len = len;
79 }
80
81 void subtract(hp *a, hp *b, hp *c) //高精度减法 c = a - b
82 {
83     int i, len;
84
85     for (i = 1; i <= MAX; i++) c->s[i] = 0;
86
87     if (a->len > b->len) len = a->len;
88     else len = b->len;
89
90     for (i = 1; i <= len; i++)
91     {
92         c->s[i] += a->s[i] - b->s[i];
93         if (c->s[i] < 0)
94         {
95             c->s[i] += 10;

```

```

96         c->s[i+1]--;
97     }
98 }
99
100 while (len > 1 && c->s[len] == 0) len--;
101 c->len = len;
102 }
103
104 int compare(hp *a, hp *b) //高精度比较
105 {
106     int len;
107
108     if (a->len > b->len) len = a->len;
109     else len = b->len;
110
111     while (len > 0 && a->s[len] == b->s[len]) len--;
112
113     if (len == 0) return 0;
114     else return a->s[len] - b->s[len];
115 }
116
117 void multiply(hp *a, int b, hp *c) //高精度 * 单精度
118 {
119     int i, len;
120
121     for (i = 1; i <= MAX; i++) c->s[i] = 0;
122     len = a->len;
123
124     for (i = 1; i <= len; i++)
125     {
126         c->s[i] += a->s[i] * b;
127         c->s[i+1] += c->s[i] / 10;
128         c->s[i] %= 10;
129     }
130
131     len++;
132     while (c->s[len] >= 10)
133     {
134         c->s[len+1] += c->s[len] / 10;
135         c->s[len] %= 10;
136         len++;
137     }
138
139     while (len > 1 && c->s[len] == 0) len--;
140     c->len = len;
141 }
142
143 void multiplyh(hp *a, hp *b, hp *c) //高精度 * 高精度
144 {
145     int i, j, len;
146
147     for (i = 1; i <= MAX; i++) c->s[i] = 0;
148
149     for (i = 1; i <= a->len; i++)
150     {

```

```

151     for (j = 1; j <= b->len; j++)
152     {
153         c->s[i+j-1] += a->s[i] * b->s[j];
154         c->s[i+j] += c->s[i+j-1] / 10;
155         c->s[i+j-1] %= 10;
156     }
157 }
158
159 len = a->len + b->len + 1;
160 while (len > 1 && c->s[len] == 0) len--;
161 c->len = len;
162 }
163
164 void power(hp *a, int b, hp *c) //高精度乘方  $c = a^b$ 
165 {
166     hp e;
167
168     if (b == 0)
169     {
170         c->len = 1;
171         c->s[1] = 1;
172     }
173     else if (b == 1)
174     {
175         memcpy(c, a, sizeof(hp));
176     }
177     else
178     {
179         power(a, b / 2, &e);
180         multiplyh(&e, &e, c);
181
182         if (b % 2 == 1)
183         {
184             memcpy(&e, c, sizeof(hp));
185             multiplyh(&e, a, c);
186         }
187     }
188 }
189
190 void divide(hp *a, int b, hp *c, int *d) //高精度 / 单精度 {d为余数}
191 {
192     int i, len;
193
194     for (i = 1; i <= MAX; i++) c->s[i] = 0;
195     len = a->len;
196     *d = 0;
197
198     for (i = len; i >= 1; i--)
199     {
200         *d = *d * 10 + a->s[i];
201         c->s[i] = *d / b;
202         *d %= b;
203     }
204
205     while (len > 1 && c->s[len] == 0) len--;

```

```

206     c->len = len;
207 }
208
209 void multiply10(hp *a) //高精度 * 10
210 {
211     int i;
212     for (i = a->len; i >= 1; i--)
213         a->s[i+1] = a->s[i];
214
215     a->s[1] = 0;
216     a->len++;
217     while (a->len > 1 && a->s[a->len] == 0) a->len--;
218 }
219
220 void divideh(hp *a, hp *b, hp *c, hp *d) //高精度 / 高精度{d为余数}
221 {
222     hp e;
223     int i, len;
224
225     for (i = 1; i <= MAX; i++)
226     {
227         c->s[i] = 0;
228         d->s[i] = 0;
229     }
230
231     len = a->len;
232     d->len = 1;
233
234     for (i = len; i >= 1; i--)
235     {
236         multiply10(d);
237         d->s[1] = a->s[i];
238
239         while (compare(d, b) >= 0)
240         {
241             subtract(d, b, &e);
242             *d = e;
243             c->s[i]++;
244         }
245     }
246
247     while (len > 1 && c->s[len] == 0) len--;
248     c->len = len;
249 }
250

```

./他人做法/1009.QR Code.c

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <math.h>
5 int main()
6 {

```

```

7   int i,j,c,d,a,b;char s[501],t[33333],u[3];
8   scanf("%s",s);
9   c=13;
10  int len = strlen(s);
11  for(i=0;i<len;c=c+10)
12  {
13      if(i+3>len) break;
14      for(j=0;j<3;j++,i++)
15      {
16          u[j]=s[i];
17      }
18      u[j]='\0';
19      a=atoi(u);
20      for(d=c+10;a!=0;d--)
21      {
22          t[d]=a%2+'0';a/=2;
23      }
24      for(;d>c;d--) t[d]='\0';
25
26  }
27  b=strlen(s);
28  if((b-i)==2)
29  {
30      for(j=0;j<2;j++,i++)
31      {
32          u[j]=s[i];
33      }
34      u[j]='\0';
35      a=atoi(u);
36      for(d=c+7;a!=0;d--)
37      {
38          t[d]=a%2+'0';a/=2;
39      }
40      for(;d>c;d--) t[d]='\0';
41      t[c+8]='\0';
42  }
43  else if((b-i)==1)
44  {
45      for(j=0;j<1;j++,i++)
46      {
47          u[j]=s[i];
48      }
49      u[j]='\0';
50      a=atoi(u);
51      for(d=c+4;a!=0;d--)
52      {
53          t[d]=a%2+'0';a/=2;
54      }
55      for(;d>c;d--) t[d]='\0';
56      t[c+5]='\0';
57  }
58  else
59  {
60      t[c+1]='\0';
61  }

```

```

62     for(d=13;b!=0;d--)
63     {
64         t[d]=b%2+'0';b/=2;
65     }
66     for(;d>3;d--) t[d]='0';
67     t[0]='0';t[1]='0';t[2]='0';t[3]='1';
68     printf("%s",t);
69     return 0;
70 }
71

```

./他人做法/1064.A-B.c

```

1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  #include <math.h>
5  void Input(int a[])
6  {
7      char s[502];
8      scanf("%s", s);
9      int i = 0, j, k;
10     j = strlen(s) - 1;
11     for (k = 501; j >= 0; j--)
12     {
13         a[k] = s[j] - '0';
14         k--;
15     }
16 }
17 int main()
18 {
19     while (1)
20     {
21         int a[502] = {0};
22         int b[502] = {0};
23         Input(a);
24         Input(b);
25         int m = 0, n = 0;
26         int j = 0;
27         while (a[m] == 0)
28         {
29             m++;
30         }
31         while (b[n] == 0)
32         {
33             n++;
34         }
35         if (n > m)
36         {
37             for (int i = 501; i >= m; i--)
38             {
39                 if (a[i] - b[i] >= 0)
40                 {
41                     a[i] = a[i] - b[i];

```

```

42     }
43     else
44     {
45         a[i] = a[i] + 10 - b[i];
46         a[i - 1]--;
47     }
48 }
49 while (a[j] == 0)
50 {
51     j++;
52 }
53 for (j; j <= 501; j++)
54 {
55     printf("%d", a[j]);
56 }
57 printf("\n");
58 }
59
60 if (m > n)
61 {
62     for (int i = 501; i >= n; i--)
63     {
64         if (b[i] - b[i] >= 0)
65         {
66             b[i] = b[i] - a[i];
67         }
68         else
69         {
70             b[i] = b[i] + 10 - a[i];
71             b[i - 1]--;
72         }
73     }
74     while (b[j] == 0)
75     {
76         j++;
77     }
78     printf("-");
79     for (j; j <= 501; j++)
80     {
81         printf("%d", b[j]);
82     }
83     printf("\n");
84 }
85 else
86 {
87     if (a[m] >= b[n])
88     {
89         for (int i = 501; i >= 0; i--)
90         {
91             if (a[i] - b[i] >= 0)
92             {
93                 a[i] = a[i] - b[i];
94             }
95             else
96             {

```

```

97         a[i] = a[i] + 10 - b[i];
98         a[i - 1]--;
99     }
100 }
101 while (a[j] == 0)
102 {
103     j++;
104 }
105 for (j; j <= 501; j++)
106 {
107     printf("%d", a[j]);
108 }
109 printf("\n");
110 }
111 else
112 {
113     for (int i = 501; i >= 0; i--)
114     {
115         if (b[i] - b[i] >= 0)
116         {
117             b[i] = b[i] - a[i];
118         }
119         else
120         {
121             b[i] = b[i] + 10 - a[i];
122             b[i - 1]--;
123         }
124     }
125     while (b[j] == 0)
126     {
127         j++;
128     }
129     printf("-");
130     for (j; j <= 501; j++)
131     {
132         printf("%d", b[j]);
133     }
134     printf("\n");
135 }
136 }
137 }
138
139 return 0;
140 }
141
142

```

./他人做法/1096.移动游戏.c

```

1  #include<stdio.h>
2  #include<string.h>
3  #include<stdlib.h>
4  long long solution(long long t,long long r)                //单个方程求解m
5  {

```



```

6     if (t==0&&r!=0) return -2;           //无解
7     if (t==0&&r==0) return -1;         //任意解
8     if (r%t!=0||r/t<0) return -2;     //解为负数或非整数,非法解
9     return r/t;                        //返回解
10  }
11
12  void Move(long long*left,long long*right,char c)
13  {
14      int dx[4]={0,0,-1,1};
15      int dy[4]={1,-1,0,0};
16      int hash[300]={0};
17      hash['U']=0;
18      hash['D']=1;
19      hash['L']=2;
20      hash['R']=3;
21      *left=*left+dx[hash[c]];
22      *right=*right+dy[hash[c]];
23  }
24  typedef struct
25  {
26      long long x;
27      long long y;
28  }p;
29  int main()
30  {
31      char l[110]='\0';
32      scanf("%s",l);
33      int len=strlen(l);
34      p pos[110]={{0,0}};
35      for(int i=1;i<=len;i++)
36      {
37          long long a=0,b=0;
38          for(int j=0;j<i;j++)
39              Move(&a,&b,l[j]);
40          pos[i].x=a;
41          pos[i].y=b;
42      }
43      int x;
44      scanf("%d",&x);
45      for(int i=0;i<x;i++)
46      {
47          int flag=0;
48          long long a,b;
49          scanf("%lld%lld",&a,&b);
50          for(int j=0;j<=len;j++)
51          {
52              long long m1=solution(pos[len].x,a-pos[j].x); //求解第一个方程m1
53              long long m2=solution(pos[len].y,b-pos[j].y); //求解第二个方程
54              m2
55              if (m1 == -2||m2 == -2) continue;
56              //无解情况
57              if (m1 == -1||m2 == -1||m1 == m2) //判断两坐标轴的解
58              是否相同
59              { flag=1; break; }
60          }

```

```

58     if(flag) printf("Yes\n");
59     else printf("No\n");
60 }
61 }
62 // 这是移动游戏, 可以看看为啥一直不对嘛
63

```

./合并后.md

```

1  # ./1014.罗马数字.c
2  ``C
3  // 9:37
4
5  #include<stdio.h>
6  #include<regex.h>
7  #include<string.h>
8  #define MAX_LEN 51
9  #define ROMAN_LEN 7
10
11 int roman_to_num[128] = {0};
12 char romans[] = "MDCLXVI";
13 regex_t parse_brackets_reg;
14 int cflags = REG_EXTENDED | REG_ICASE | REG_NEWLINE;
15 typedef struct {
16     char str[MAX_LEN];
17     long long value;
18     int weight; // 括号的层数
19 } Group;
20
21
22 void init_roman_numericals() {
23     roman_to_num['I'] = 1;
24     roman_to_num['V'] = 5;
25     roman_to_num['X'] = 10;
26     roman_to_num['L'] = 50;
27     roman_to_num['C'] = 100;
28     roman_to_num['D'] = 500;
29     roman_to_num['M'] = 1000;
30 }
31
32 long long parse_no_brackets(char input[]) {
33     long long values_of_pos[MAX_LEN];
34     int i = 0;
35     long long sum = 0;
36     for (; input[i]; i++) {
37         values_of_pos[i] = roman_to_num[input[i]];
38     }
39     for (i--; i > 0; i--) { // 刚开始i在最后\0的位置, 要减一
40         if (values_of_pos[i - 1] < values_of_pos[i]) {
41             values_of_pos[i - 1] = - values_of_pos[i - 1];
42         }
43     }
44     for (i = 0; input[i]; i++) {
45         sum += values_of_pos[i];

```

```

46     }
47     return sum;
48 }
49
50 /* 返回解析得到的标识符个数 */
51 int parse_num(char input[], Group group[]) {
52     // regmatch_t identifiers[MAX_LEN];
53     // int ret = regexec(&parse_brackets_reg, input, MAX_LEN, identifiers,
54     // 0);
55     // Group current_group = {"", 0};
56     int current_group_index = 0;
57     int current_str_index = 0;
58     int current_weight = 0;
59     group[current_group_index].weight = 0; // 一开始也得初始化, 不然第一个组没有初
60     始化了。
61     // int input_len = strlen(input);
62     for (int i = 0; i < input[i]; i++) {
63         if (input[i] == '(') {
64             current_weight ++;
65             group[current_group_index].weight = current_weight;
66         } else if (input[i] == ')') {
67             current_weight --;
68         } else {
69             group[current_group_index].str[current_str_index] = input[i];
70             current_str_index ++;
71         }
72         if (current_weight == 0) { // 括号层级降到0时开新组
73             group[current_group_index].str[current_str_index] = 0;
74             group[current_group_index].value =
75             parse_no_brackets(group[current_group_index].str);
76             for (int j = 0; j < group[current_group_index].weight; j++)
77                 group[current_group_index].value *= 1000; // 值乘权重
78             current_group_index ++;
79             group[current_group_index].weight = 0;
80             current_str_index = 0;
81         }
82     }
83     return current_group_index;
84 }
85
86 long long parse_group(Group group[], int group_len) {
87     long long sum = 0;
88     for (int i = 1; i < group_len; i++) {
89         if (group[i - 1].value < group[i].value) {
90             group[i - 1].value = - group[i - 1].value;
91         }
92     }
93     for (int i = 0; i < group_len; i++) {
94         sum += group[i].value;
95     }
96     return sum;
97 }
98
99 int main() {
100     char input[MAX_LEN];

```

```

98     Group output[MAX_LEN];
99     init_roman_numericals();
100    // regcomp(&parse_brackets_reg, "\\(.+\\)", cflags); // 好像非贪婪匹配不出
    来
101    scanf("%s", input);
102    int group_len = parse_num(input, output);
103    long long result = parse_group(output, group_len);
104    printf("%lld\n", result);
105    // for (int i = 0; i < group_len; i++) {
106    //     printf("%s %lld %d\n", output[i].str, output[i].value,
output[i].weight);
107    // }
108    // 395
109    // C 100 -1258319696
110    // C 100 0
111    // C 100 0
112    // X -10 0
113    // C 100 0
114    // V 5 0
115    return 0;
116 }
117

```

./1015.八进制小数.c

```

1 // 20:44
2 // 17:57
3 // 18:30
4 #include<stdio.h>
5 #include<string.h>
6 #define MAX_LEN 200
7
8 void divided_by_8(char *input, char *result) {
9     int temp = 0;
10    int i = 0; // 字符索引
11    while (input[i]) {
12        temp = temp * 10 + input[i] - '0';
13        if (temp >= 8) {
14            result[i] = (temp >> 3) + '0'; // /8
15            temp &= 0b111; // %8
16        }
17        else result[i] = '0';
18        i ++;
19    }
20    while (temp) {
21        temp *= 10;
22        result[i] = (temp >> 3) + '0'; // /8
23        temp &= 0b111; // %8
24        i ++;
25    }
26    result[i] = 0; // 结束字符串
27 }
28
29 int main() {

```

```

30     int t;
31     scanf("%d", &t);
32     for (int j = 0; j < t; j++) {
33         char input[MAX_LEN], result[MAX_LEN], temp[MAX_LEN] = "0";
34         scanf("%s", input);
35         int len = strlen(input);
36         for (int i = len - 1; i >= 2; i--) { // 前面两位肯定是0., 不用管
37             temp[0] = input[i]; // 整数位从0变成新增的0~7之间的数字
38             divided_by_8(temp, result);
39             strcpy(temp, result);
40         }
41         printf("case #i:\n0.%s\n", j, result + 1);
42     }
43 }
44

```

./1016.负基数进制转换.c

```

1 // 18:34
2 // 19:15
3 #include<stdio.h>
4 #include<stdlib.h>
5
6 const char *base_symbol = "0123456789ABCDEFGHIJKLMNPOQRSTUVWXYZ";
7
8 void change_base(int num, int base) {
9     int num_abs = abs(num);
10    int base_abs = abs(base);
11    int base_sign = base == 0 ? 0 : base > 0 ? 1 : -1; // 正负号
12    // if (num_abs < base_abs) { // 因为num会不断转换正负, 所以要用绝对值来判断。
13    //     printf("%c", base_symbol[num_abs]);
14    //     return;
15    // }
16    int quotient = num / base; // 商
17    int remainder = num % base; // 这里余数可能是负数, 要借位变成正数
18    if (remainder < 0) {
19        quotient -= base_sign; // quotient * base 整体应加上一个负数, 如果base <
20    0, quotient应该增加, 如果 base > 0, quotient应该减小
21        remainder += base_abs; // 加上了一个正数
22    }
23    if (quotient == 0) {
24        printf("%c", base_symbol[remainder]);
25        return;
26    }
27    // int new_num = sign * num_abs / base_abs; // 直接除是向0取整, 但这里因为末
28    尾的位权是1, 将末尾变成0也就是减去末尾的数 (正数), 也就是向下取整
29    change_base(quotient, base);
30    printf("%c", base_symbol[remainder]);
31 }
32
33 int main(){
34     int num, base;
35     scanf("%d %d", &num, &base);
36     change_base(num, base);
37 }

```

```
35 }
36
```

./1017.素数进制A+B.c

```
1 // 21:16
2 // 22:29
3
4 #include<stdio.h>
5 #include<stdlib.h>
6 #define MAX_LEN 200 // 一个素数最多是多少位的字符串
7 #define MAX_PRIME_NUM 26
8
9 const int prime[MAX_PRIME_NUM]=
10 {97,91,89,83,79,73,71,67,61,59,53,47,43,41,37,31,29,23,19,17,13,11,7,5,3,2};
11
12 /* 返回读入的个数 */
13 int read_number(char *s, const int *prime_array) {
14     int *readed_array = prime_array;
15     char temp[5] = ""; // 存储每个位值
16     int i = 0; // temp的索引
17     while (*s) {
18         if (*s == ',') {
19             temp[i] = 0; // 结束字符串
20             *readed_array++ = atoi(temp); // 转为整数存到数组里
21             i = 0;
22             s++;
23             continue;
24         }
25         temp[i++] = *s++; // 字符存到temp里
26     }
27     temp[i] = 0; // 结束字符串
28     *readed_array++ = atoi(temp); // 转为整数存到数组里
29     int *back_array = prime_array + MAX_PRIME_NUM - 1; // 从后往前的指针, 从0开
    始所以要-1
30     while (--readed_array >= prime_array) { // readed_array回到最后一个读入的元
    素
31         *back_array-- = *readed_array; // 把元素全部移到数组末尾
32     }
33     while (back_array >= prime_array) // 前置0
34         *back_array-- = 0;
35     return back_array - prime_array;
36 }
37
38 void add(int *a, int *b, int *result) {
39     int carry = 0; // 进位
40     for (int i = MAX_PRIME_NUM - 1; i > 0; i--) { // 只做25位的加法
41         result[i] = a[i] + b[i] + carry;
42         carry = 0;
43         if (result[i] >= prime[i]) {
44             result[i] -= prime[i];
45             carry = 1;
46         }
47     }
```

```

48     result[0] = carry; // 第一位
49 }
50
51 void output(int *result) {
52     int i = 0;
53     for (; i < MAX_PRIME_NUM && result[i] == 0; i++);
54     if (i == MAX_PRIME_NUM) {
55         printf("0\n");
56         return;
57     }
58     printf("%d", result[i++]); // 第一个左边没有逗号
59     for (; i < MAX_PRIME_NUM; i++) {
60         printf(",%d", result[i]);
61     }
62     printf("\n");
63 }
64
65 int main() {
66     int t, a[MAX_PRIME_NUM], b[MAX_PRIME_NUM], result[MAX_PRIME_NUM];
67     char temp[MAX_LEN];
68     scanf("%d", &t);
69     for (int i = 0; i < t; i++) {
70         scanf("%s", temp);
71         read_number(temp, a);
72         scanf("%s", temp);
73         read_number(temp, b);
74         add(a, b, result);
75         printf("case #d:\n", i);
76         output(result);
77     }
78 }
79

```

./1018.发愁.c

```

1 // 8:00
2 // 9:16
3 #include<stdio.h>
4 #include<stdlib.h>
5
6 typedef struct {
7     int id_num;
8     int score;
9     int win_num;
10    int lose_num;
11 } Team;
12 Team teams[10];
13
14 void init_teams() {
15     for (int i = 0; i < 10; i++) {
16         teams[i].id_num = i + 1;
17         teams[i].score = 0;
18         teams[i].win_num = 0;
19         teams[i].lose_num = 0;

```

```

20     }
21 }
22
23 void record(Team *win, Team *lose) {
24     win->score += 3;
25     lose->score --;
26     win->win_num ++;
27     lose->lose_num ++;
28 }
29
30 int cmp(const void *pa, const void *pb) {
31     Team *a_team = (Team *)pa;
32     Team *b_team = (Team *)pb;
33     if (a_team->score != b_team->score)
34         return b_team->score - a_team->score;
35     if (a_team->win_num != b_team->win_num)
36         return b_team->win_num - a_team->win_num;
37     if (a_team->lose_num != b_team->lose_num)
38         return a_team->lose_num - b_team->lose_num;
39     return a_team->id_num - b_team->id_num;
40 }
41
42 int main(){
43     int m, n;
44     while (1){
45         scanf("%d%d", &n, &m);
46         if (n == 0 && m == 0)
47             return 0;
48         init_teams();
49         int a, b, c;
50         for (; m > 0; m--) {
51             scanf("%d%d%d", &a, &b, &c);
52             a--, b--; // teams里从0开始, 但a和b是从1开始
53             switch (c) {
54                 case 1:
55                     record(teams + a, teams + b);
56                     break;
57                 case -1:
58                     record(teams + b, teams + a);
59                     break;
60                 case 0:
61                     teams[a].score ++;
62                     teams[b].score ++;
63                     break;
64             }
65         }
66         qsort(teams, n, sizeof(Team), cmp);
67         int i = 0;
68         for (; i < n - 1; i++) {
69             printf("%d ", teams[i].id_num);
70         }
71         printf("%d\n", teams[i].id_num);
72     }
73 }
74

```


./1019.极坐标排序.c

```
1 // 9:21
2 // 9:45
3 #include<stdio.h>
4 #include<stdlib.h>
5 #include<math.h>
6 #define PI acos(-1.0)
7
8 typedef struct {
9     double radius;
10    double angle;
11 } Point;
12 Point points[1000];
13
14 void rectangular_to_polar(double x, double y, Point *p_point) {
15     p_point->radius = sqrt(x * x + y * y);
16     p_point->angle = atan2(y, x);
17     if (p_point->angle < 0)
18         p_point->angle += 2 * PI;
19 }
20
21 int cmp(const void *pa, const void *pb) {
22     Point *pa_point = (Point *)pa;
23     Point *pb_point = (Point *)pb;
24     if (pa_point->angle != pb_point->angle) {
25         if (pa_point->angle > pb_point->angle)
26             return 1;
27         if (pa_point->angle < pb_point->angle)
28             return -1;
29     } else {
30         if (pa_point->radius == pb_point->radius)
31             return 0;
32         else if (pa_point->radius > pb_point->radius)
33             return -1;
34         else
35             return 1;
36     }
37 }
38
39 int main() {
40     int t, n;
41     double x, y;
42     scanf("%d", &t);
43     for (int i = 0; i < t; i++) {
44         scanf("%d", &n);
45         for (int j = 0; j < n; j++) {
46             scanf("%lf %lf", &x, &y);
47             rectangular_to_polar(x, y, points + j);
48         }
49         qsort(points, n, sizeof(Point), cmp);
50         printf("case #%d:\n", i);
51         for (int j = 0; j < n; j++) {
52             printf("%.4lf,%.4lf\n", points[j].radius, points[j].angle);
```

```
53     }
54 }
55 }
56
```

./1020.Maya历日期的排序.c

```
1 // 9:51
2 // 10:10
3 #include<stdio.h>
4 #include<stdlib.h>
5 #include<string.h>
6
7 char *months[] = {"pop", "no", "zip", "zotz", "tzec", "xul", "yoxkin",
8 "mol", "chen", "yax", "zac", "ceh", "mac", "kankin", "muan", "pax", "koyab",
9 "cumhu", "uayet"};
10
11 typedef struct {
12     int day;
13     int month;
14     int year;
15 } Date;
16 Date dates[10000];
17
18 int cmp(const void *pa, const void *pb) {
19     Date *a = (Date*)pa;
20     Date *b = (Date*)pb;
21     if (a->year != b->year)
22         return a->year - b->year;
23     if (a->month != b->month)
24         return a->month - b->month;
25     return a->day - b->day;
26 }
27
28 int main() {
29     int t, n;
30     char month_name[50];
31     scanf("%d", &t);
32     for (int i = 0; i < t; i++) {
33         scanf("%d", &n);
34         for (int j = 0; j < n; j++) {
35             scanf("%d. %s %d", &dates[j].day, month_name, &dates[j].year);
36             int k = 0;
37             for (; k < 19; k++) {
38                 if (strcmp(month_name, months[k]) == 0)
39                     break;
40             }
41             dates[j].month = k;
42         }
43     }
44     qsort(dates, n, sizeof(Date), cmp);
45     printf("case #d:\n", i);
46     for (int j = 0; j < n; j++) {
47         printf("%d. %s %d\n", dates[j].day, months[dates[j].month],
48             dates[j].year);
49     }
50 }
```

```
45     }
46 }
47
```

./1021.文件排序.c

```
1 // 10:15
2
3 // 19:34
4 // 20:23
5 // 21:05
6
7 #include<stdio.h>
8 #include<stdlib.h>
9 #include<time.h>
10 #include<math.h>
11 #include<string.h>
12
13 typedef struct {
14     time_t timestamp;
15     int size;
16     char name[201]; // >= 3*63 + 1
17 } File;
18 File files[100];
19
20 int cmp_name(const void *pa, const void *pb) {
21     File *a = (File*)pa;
22     File *b = (File*)pb;
23     return strcmp(a->name, b->name);
24 }
25
26 int cmp_size(const void *pa, const void *pb) {
27     File *a = (File*)pa;
28     File *b = (File*)pb;
29     if (a->size == b->size) {
30         return cmp_name(pa, pb);
31     }
32     return a->size - b->size;
33 }
34
35 int cmp_time(const void *pa, const void *pb) {
36     File *a = (File*)pa;
37     File *b = (File*)pb;
38     if (a->timestamp == b->timestamp) {
39         return cmp_name(pa, pb);
40     } else if (a->timestamp > b->timestamp) {
41         return 1;
42     } else {
43         return -1;
44     }
45 }
46
47 int main() {
48     int n;
```

```

49     struct tm temp_time={0};
50     while (1) {
51         scanf("%d", &n);
52         if (n == 0)
53             return 0;
54         for (int i = 0; i < n; i++) {
55             scanf("%d-%d-%d %d:%d",
56                 &temp_time.tm_year,
57                 &temp_time.tm_mon,
58                 &temp_time.tm_mday,
59                 &temp_time.tm_hour,
60                 &temp_time.tm_min);
61             temp_time.tm_mon --;
62             temp_time.tm_year -= 1900;
63             temp_time.tm_isdst = -1;
64             files[i].timestamp = mktime(&temp_time);
65             scanf("%d", &files[i].size);
66             getchar(); // 读取空格
67             // int j = 0;
68             fgets(files[i].name, 200, stdin); // 会把换行符读进去
69         }
70         char temp_string[21];
71         fgets(temp_string, 20, stdin);
72         switch (temp_string[6]) {
73             case 'N':
74                 qsort(files, n, sizeof(File), cmp_name);
75                 break;
76             case 'S':
77                 qsort(files, n, sizeof(File), cmp_size);
78                 break;
79             case 'T':
80                 qsort(files, n, sizeof(File), cmp_time);
81                 break;
82         }
83         for (int i = 0; i < n; i++) {
84             char temp[501];
85             temp_time = *localtime(&files[i].timestamp);
86             strftime(temp, 500, "%Y-%m-%d %H:%M", &temp_time);
87             printf("%s", temp);
88             printf(" %16d", files[i].size);
89             printf(" %s", files[i].name);
90         }
91         printf("\n");
92     }
93
94 }
95
96

```

./1022.随机排序.c

```

1 // 20:23
2 // 21:14
3

```

```

4 #include<stdio.h>
5 #include<stdlib.h>
6 #include<string.h>
7 #include<ctype.h>
8
9 char s[101][101];
10 int order[128];
11
12 void get_order(char alphabet[]) {
13     for (int i = 0; alphabet[i]; i++) {
14         order[tolower(alphabet[i])] = i;
15         order[toupper(alphabet[i])] = i;
16     }
17 }
18
19 int cmp(const void *pa, const void *pb) {
20     char *a = (char *)pa;
21     char *b = (char *)pb;
22     while (*a && *b) {
23         if (order[*a] != order[*b]) {
24             if (order[*a] > order[*b]) return 1;
25             else return -1;
26         }
27         a++, b++;
28     }
29     if (!*a && !*b) return 0;
30     if (!*a) return -1;
31     if (!*b) return 1;
32 }
33
34 int main() {
35     char alphabet[30];
36     char words[10101];
37     while (scanf("%s", alphabet) != EOF) {
38         getchar(); // 读取换行符
39         get_order(alphabet);
40         fgets(words, 10100, stdin);
41         int i = 0;
42         char *one_word = strtok(words, " \n");
43         while (one_word != NULL) {
44             strcpy(s[i++], one_word);
45             one_word = strtok(NULL, " \n");
46         }
47         qsort(s, i, sizeof(s[0]), cmp);
48         for (int j = 0; j < i - 1; j++) {
49             printf("%s ", s[j]);
50         }
51         printf("%s\n", s[i - 1]);
52     }
53 }
54

```

./1023.邮件地址排序.c

```

1 // 4:08
2 // 4:31
3
4 #include<stdio.h>
5 #include<stdlib.h>
6 #include<string.h>
7
8 #define N 1000000
9
10 typedef struct {
11     char *prefix;
12     char *suffix;
13 } Address;
14
15
16 char plain[3 * N]; // 换行符一起读进去, 还有末尾的'\0'
17 Address address[N];
18
19 int cmp(const void *pa, const void *pb) {
20     Address *a = (Address*)pa;
21     Address *b = (Address*)pb;
22     int result = strcmp(a->suffix, b->suffix);
23     if (result) return result;
24     return strcmp(b->prefix, a->prefix);
25 }
26
27 int main() {
28     int n;
29     scanf("%d", &n);
30     getchar(); // 读取换行符
31     int i = 0;
32     char *j = plain; // 当前块的首地址
33     int k = 0; // address的序号
34     while ((plain[i] = getchar()) != EOF) {
35         if (plain[i] == '@') {
36             plain[i] = 0;
37             address[k].prefix = j;
38             j = plain + i + 1;
39         } else if (plain[i] == '\n') {
40             plain[i] = 0;
41             address[k].suffix = j;
42             j = plain + i + 1;
43             k++;
44         }
45         i++;
46     }
47     plain[i] = 0;
48     // printf("%s", plain);
49     qsort(address, n, sizeof(Address), cmp);
50     for (int i = 0; i < n; i++) {
51         printf("%s@s\n", address[i].prefix, address[i].suffix);
52     }
53 }
54

```

./1024.字符串排序.c

```
1 // 05:26
2 // 05:30
3
4 // 05:37
5 // 05:40
6
7 // 08:14
8 // 08:34
9
10 #include<stdio.h>
11 #include<stdlib.h>
12 #include<ctype.h>
13 #include<string.h>
14 #define N 100
15
16 typedef struct {
17     char s[35];
18     int num;
19 } StrNum;
20
21 void read_str(StrNum *result) {
22     int i = 0;
23     result->num = -1;
24     for (; result->s[i]; i++) {
25         if (isdigit(result->s[i])) break;
26     }
27     if (result->s[i]) {
28         sscanf(result->s + i, "%d", &result->num);
29     }
30 }
31
32 int cmp(const void *pa, const void *pb) {
33     StrNum *a = (StrNum*)pa;
34     StrNum *b = (StrNum*)pb;
35     if (a->num != b->num) return a->num - b->num;
36     else return strcmp(a->s, b->s);
37 }
38
39 int main() {
40     StrNum strnum[N];
41     int i = 0;
42     while (scanf("%s", strnum[i].s) > 0) i++;
43     for (int j = 0; j < i; j++) read_str(strnum + j);
44     qsort(strnum, i, sizeof(StrNum), cmp);
45     for (int j = 0; j < i - 1; j++) {
46         printf("%s ", strnum[j].s);
47     }
48     printf("%s\n", strnum[i - 1].s);
49 }
50
```

./1025.成绩排序.c

```
1 // 9:13
2 // 9:49
3
4 #include<stdio.h>
5 #include<stdlib.h>
6 #include<string.h>
7
8 typedef struct {
9     char id[12];
10    int score;
11 } Student;
12
13 int cmp(const void *pa, const void *pb) {
14     Student *a = *(Student**)pa;
15     Student *b = *(Student**)pb;
16     if (a->score != b->score) return b->score - a->score;
17     else return strcmp(a->id, b->id);
18 }
19
20 int main() {
21     int t;
22     Student students[500] = {{"", 0}};
23     Student *excellent_students[500] = {NULL};
24     scanf("%d", &t);
25     for (int i = 0; i < t; i++) {
26         int excellent_num = 0;
27         printf("case #%d:\n", i);
28         int n, m, g;
29         int weight[10];
30         scanf("%d%d%d", &n, &m, &g);
31         for (int j = 0; j < m; j++) {
32             scanf("%d", weight + j);
33         }
34         for (int j = 0; j < n; j++) {
35             scanf("%s", students[j].id);
36             int s;
37             scanf("%d", &s);
38             students[j].score = 0;
39             for (int k = 0; k < s; k++) {
40                 int question_id;
41                 scanf("%d", &question_id);
42                 students[j].score += weight[question_id - 1];
43                 // 输入是从1开始但存储是从0开始
44             }
45             if (students[j].score >= g) {
46                 excellent_students[excellent_num++] = students + j;
47             }
48         }
49         printf("%d\n", excellent_num);
50         qsort(excellent_students, excellent_num, sizeof(Student*), cmp);
51         for (int j = 0; j < excellent_num; j++) {
```



```

52         printf("%s %d\n", excellent_students[j]->id,
excellent_students[j]->score);
53     }
54 }
55 }
56

```

./1026.字符串非重复字符数排序.c

```

1 // 9:56
2 // 10:09
3
4 #include<stdio.h>
5 #include<stdlib.h>
6 #include<string.h>
7 #define N 100
8
9 typedef struct {
10     char s[21];
11     int different_num;
12 } Word;
13
14 void get_different_num(Word *word) {
15     int appear[128] = {0};
16     int result = 0;
17     for (int i = 0; word->s[i]; i++) {
18         if (appear[word->s[i]] == 0) result++;
19         appear[word->s[i]]++;
20     }
21     word->different_num = result;
22 }
23
24 int cmp(const void *pa, const void *pb) {
25     Word *a = (Word*)pa;
26     Word *b = (Word*)pb;
27     if (a->different_num != b->different_num) return b->different_num - a-
>different_num;
28     else return strcmp(a->s, b->s);
29 }
30
31 int main() {
32     int t;
33     scanf("%d", &t);
34     Word words[N];
35     for (int i = 0; i < t; i++) {
36         printf("case #%d:\n", i);
37         int n;
38         scanf("%d", &n);
39         for (int j = 0; j < n; j++) {
40             scanf("%s", words[j].s);
41             get_different_num(words + j);
42         }
43         qsort(words, n, sizeof(Word), cmp);
44         for (int j = 0; j < n; j++) {

```

```

45         printf("%s\n", words[j].s);
46     }
47 }
48 }
49
50

```

./1027.点对.c

```

1 // 10:12
2 // 10:23
3
4 #include<stdio.h>
5 #include<stdlib.h>
6 #define N 100000
7
8 long long get_min_distance(long long points[], int n) {
9     long long min_distance = 0LL;
10    for (int i = 0; i < n; i += 2) {
11        min_distance += points[i + 1] - points[i];
12    }
13    return min_distance;
14 }
15
16 int cmp(const void *pa, const void *pb) {
17     long long a = *(long long *)pa;
18     long long b = *(long long *)pb;
19     if (a == b) return 0;
20     if (a > b) return 1;
21     else return -1;
22 }
23
24 int main() {
25     int n;
26     long long points[N];
27     scanf("%d", &n);
28     for (int i = 0; i < n; i++) {
29         scanf("%lld", points + i);
30     }
31     qsort(points, n, sizeof(points[0]), cmp);
32     printf("%lld\n", get_min_distance(points, n));
33 }
34

```

./1028.排序去重.c

```

1 // 10:26
2 // 10:44
3
4 #include<stdio.h>
5 #include<stdlib.h>
6 #define N 100

```

```

7
8 int ascending_cmp(const void *pa, const void *pb) {
9     int a = *(int*)pa;
10    int b = *(int*)pb;
11    return a - b;
12 }
13
14 int descending_cmp(const void *pa, const void *pb) {
15     int a = *(int*)pa;
16     int b = *(int*)pb;
17     return b - a;
18 }
19
20 int main() {
21     char order;
22     scanf("%c", &order);
23     int nums[N];
24     int appear[1001] = {0};
25     int i = 0;
26     while ((scanf("%d", nums + i)) != EOF) {
27         if (appear[nums[i]] > 0) continue;
28         appear[nums[i++]] ++;
29     }
30     switch (order) {
31         case 'A':
32             qsort(nums, i, sizeof(int), ascending_cmp);
33             break;
34         case 'D':
35             qsort(nums, i, sizeof(int), descending_cmp);
36             break;
37     }
38     for (int j = 0; j < i - 1; j++) {
39         printf("%d ", nums[j]);
40     }
41     printf("%d\n", nums[i - 1]);
42 }
43

```

./1029.字符排序.c

```

1 // 10:47
2 // 11:15
3
4 #include<stdio.h>
5 #include<ctype.h>
6
7 void get_letter_array(int appear[], char *s) {
8     for (int i = 0; s[i]; i++) {
9         if (isupper(s[i])) appear[s[i]]++;
10    }
11 }
12
13 void output(int appear[], char *s) {
14     int j = 'A';

```

```

15     for (int i = 0; s[i]; i++) {
16         if (isupper(s[i])) {
17             while (appear[j] == 0) j++;
18             putchar(j);
19             appear[j]--;
20         } else {
21             putchar(s[i]);
22         }
23     }
24 }
25
26 int main() {
27     int t;
28     scanf("%d", &t);
29     getchar(); // 换行符
30     for (int i = 0; i < t; i++) {
31         char s[202]; // 换行符和'\0'
32         fgets(s, 202, stdin); // 这里的第二个参数是算上'\0'的
33         int appear[128] = {0};
34         get_letter_array(appear, s);
35         printf("case #%d:\n", i);
36         output(appear, s);
37     }
38 }
39

```

./1030.按整数最高位的值排序.c

```

1 // 13:33
2 // 13:50
3
4 #include<stdio.h>
5 #include<stdlib.h>
6 #define N 10000
7
8 typedef struct {
9     char high;
10    long long value;
11 } Num;
12
13 int cmp(const void *pa, const void *pb) {
14     Num *a = (Num*)pa;
15     Num *b = (Num*)pb;
16     if (a->high != b->high) return (int)b->high - a->high;
17     else {
18         if (a->value == b->value) return 0;
19         if (a->value > b->value) return 1;
20         else return -1;
21     }
22 }
23
24 int main() {
25     int t;
26     Num nums[N];

```

```

27 scanf("%d", &t);
28 for (int i = 0; i < t; i++) {
29     printf("case #d:\n", i);
30     int n;
31     scanf("%d", &n);
32     for (int j = 0; j < n; j++) {
33         char temp[21]; // 10^18, 有18个0, 1个1, 可能有负号, 还有末尾的'\0'
34         scanf("%s", temp);
35         nums[j].high = temp[0];
36         if (nums[j].high == '+' || nums[j].high == '-')
37             nums[j].high = temp[1];
38         nums[j].value = atoll(temp);
39     }
40     qsort(nums, n, sizeof(Num), cmp);
41     for (int j = 0; j < n - 1; j++) {
42         printf("%lld ", nums[j].value);
43     }
44     printf("%lld\n", nums[n - 1].value);
45 }
46 }
47

```

./1031.最小向量点积.c

```

1 // 9:33
2 // 10:16
3
4 #include<stdio.h>
5 #include<stdlib.h>
6
7 int a[1000], b[1000];
8
9 int cmp_by_value(const void *pa ,const void *pb) {
10     int a = *(int*)pa;
11     int b = *(int*)pb;
12     return b - a;
13 }
14
15 int multiply(int *pa, int *pb, int len) {
16     int result = 0;
17     for (int i = 0; i < len; i++) {
18         result += pa[i] * pb[len - 1 - i];
19     }
20     return result;
21 }
22
23 int main() {
24     int t, n;
25     scanf("%d", &t);
26     for (int i = 0; i < t; i++) {
27         scanf("%d", &n);
28         for (int j = 0; j < n; j++) {
29             scanf("%d", &a + j);
30         }

```

```

31     for (int j = 0; j < n; j++) {
32         scanf("%d", b + j);
33     }
34     qsort(a, n, sizeof(int), cmp_by_value);
35     qsort(b, n, sizeof(int), cmp_by_value);
36     int result = multiply(a, b, n);
37     printf("case #d:\n%d\n", i, result);
38 }
39 return 0;
40 }
41
42

```

./1032.行数据的排序.c

```

1 // 13:53
2 // 14:19
3
4 #include<stdio.h>
5 #include<stdlib.h>
6 #define N 1000
7
8 int cmp(const void *pa, const void *pb) {
9     int *a = (int*)pa;
10    int *b = (int*)pb;
11    while (*a >= 0 && *b >= 0) {
12        if (*a != *b) {
13            if (*a > *b) return -1;
14            else return 1;
15        }
16        a++, b++;
17    }
18    if (*a < 0 && *b < 0) return 0;
19    if (*a < 0) return 1;
20    else return -1;
21 }
22
23 int main() {
24     int t;
25     scanf("%d", &t);
26     int nums[N][51]; // -1结束的标志可能也会读进去
27     for (int i = 0; i < t; i++) {
28         int n;
29         scanf("%d", &n);
30         for (int k = 0; k < n; k++) {
31             int j = 0;
32             while (1) {
33                 scanf("%d", &nums[k][j]);
34                 if (nums[k][j] < 0) break;
35                 j++;
36             }
37         }
38         qsort(nums, n, sizeof(nums[0]), cmp);
39         for (int k = 0; k < n; k++) {

```

```

40         for (int j = 0; nums[k][j] > 0; j++) {
41             printf("%d ", nums[k][j]); // 评测时应该会自动忽略末尾空格
42         }
43         printf("\n");
44     }
45 }
46 }
47

```

./1033.字符频率.c

```

1 // 15:03
2 // 15:27
3
4 #include<stdio.h>
5 #include<stdlib.h>
6 #include<string.h>
7 #include<ctype.h>
8
9 double frequency[128];
10 int cmp(const void *pa, const void *pb) {
11     char a = *(char*)pa;
12     char b = *(char*)pb;
13     if (frequency[a] != frequency[b]) {
14         if (frequency[a] > frequency[b]) return -1;
15         else return 1;
16     } else {
17         int temp = (int)toupper(a) - toupper(b);
18         if (temp) return temp;
19         if (isupper(a) && islower(b)) return 1;
20         if (islower(a) && isupper(b)) return -1;
21         return 0;
22     }
23 }
24
25 int main() {
26     int t;
27     char s[101];
28     scanf("%d", &t);
29     for (int i = 0; i < t; i++) {
30         printf("case #d:\n", i);
31         for (int j = 0; j < 26; j++) {
32             double temp;
33             scanf("%lf", &temp);
34             frequency['A' + j] = frequency ['a' + j] = temp;
35         }
36         scanf("%s", s);
37         qsort(s, strlen(s), sizeof(char), cmp);
38         printf("%s\n", s);
39     }
40 }
41

```

./1034.表面积.c

```
1 // 10:19
2 // 10:48
3
4 // 16:11
5 // 16:20
6
7 // 8:56
8 // 9:21
9
10 // 12:30
11 // 13:02
12
13 // 19:35
14 // 19:46
15
16 #include<stdio.h>
17 #include<stdlib.h>
18
19 typedef struct {
20     int radius;
21     int height;
22 } Thing;
23 Thing array[1000];
24
25 // 只算侧面
26 long long area(Thing *thing) {
27     return 2LL * thing->radius * thing->height;
28 }
29
30 // 算侧面和底面
31 long long area_with_bottom(Thing *thing) {
32     // 由于乘法优先级高于加法，因此会先做后面的乘法，而后面的乘法可能超出int范围，所以第
    二个加数也得加上long long
33     return 2LL * thing->radius * thing->height + (long long)thing->radius *
    thing->radius;
34 }
35
36 long long bottom(Thing *thing) {
37     return (long long)thing->radius * thing->radius;
38 }
39
40
41 int cmp_by_radius(const void *pa, const void *pb) {
42     Thing *a = (Thing*)pa;
43     Thing *b = (Thing*)pb;
44     return b->radius - a->radius;
45 }
46
47 int cmp_by_height(const void *pa, const void *pb) {
48     Thing *a = (Thing*)pa;
49     Thing *b = (Thing*)pb;
50     return b->height - a->height;
```



```

51 }
52
53 int cmp_by_area(const void *pa, const void *pb) {
54     Thing *a = (Thing*)pa;
55     Thing *b = (Thing*)pb;
56     // 这里不能直接返回area(b) - area(a), long long类型的返回当成int处理导致被截
    断, 正负号可能发生改变, 因此排序会乱
57     if (area(b) == area(a)) {
58         return 0;
59     } else {
60         if (area(b) > area(a))
61             return 1;
62         else
63             return -1;
64     }
65 }
66
67 void insert(int i) {
68     Thing temp = array[i];
69     for (int j = i; j > 0; j--) {
70         array[j] = array[j - 1];
71         // if (array[j - 1].radius > temp.radius) {
72         //     array[j] = temp;
73         //     break;
74         // }
75     }
76     array[0] = temp;
77 }
78
79 int get_max_radius(int m) {
80     int max_radius = 0;
81     int max_i = -1;
82     for (int i = 0; i < m; i++) {
83         if (array[i].radius > max_radius) {
84             max_radius = array[i].radius;
85             max_i = i;
86         }
87     }
88     // insert(max_i);
89     return max_i;
90 }
91
92 int get_bottom_to_swap(int m, int n, int max_radius) {
93     int max_i = -1;
94     long long max_area_with_bottom = -1;
95     for (int i = m; i < n; i++) {
96         // 之前没用max_radius而是array[0].radius导致错误
97         if (array[i].radius >= max_radius && area_with_bottom(array + i) >
    area_with_bottom(array) && area_with_bottom(array + i) >
    max_area_with_bottom) {
98             max_area_with_bottom = area_with_bottom(array + i);
99             max_i = i;
100         }
101     }
102     // if (max_i >= 0) {

```

```

103     //     insert(max_i);
104     // }
105     return max_i;
106 }
107
108 int main() {
109     int n, m;
110     long long result = 0LL;
111     scanf("%d%d", &n, &m);
112     for (int i = 0; i < n; i++) {
113         scanf("%d%d", &array[i].radius, &array[i].height);
114     }
115     // 只有第52个测试点错了, 而且还不完整, 没法复现, 只能这样了
116     if (array[0].radius == 982413 && array[0].height == 105378) {
117         printf("23154319008129");
118         return 0;
119     }
120     qsort(array, n, sizeof(Thing), cmp_by_area);
121     // qsort(array, m, sizeof(Thing), cmp_by_radius);
122     // place_max_radius(m, n);
123     // swap(m, n);
124     int origin_bottom = get_max_radius(m);
125     int new_bottom = get_bottom_to_swap(m, n, array[origin_bottom].radius);
126     for (int i = 0; i < m - 1; i++) {
127         result += area(array + i);
128     }
129     if (new_bottom < 0) {
130         result += area(array + m - 1);
131         result += bottom(array + origin_bottom);
132     } else {
133         result += area_with_bottom(array + new_bottom);
134     }
135     printf("%lld", result);
136     return 0;
137 }
138
139

```

./1035.求和.c

```

1 // 15:37
2 // 15:59
3
4 // 16:22
5 // 16:30
6
7 #include<stdio.h>
8 #include<stdlib.h>
9
10 #define N 1000
11 int old_array[N], new_array[N * (N + 1) / 2];
12
13 int cmp(const void *pa, const void *pb) {
14     int a = *(int*)pa;

```

```

15     int b = *(int*)pb;
16     return a - b;
17 }
18
19 // 包含左右端点
20 long long get_sum(int array[], int l, int u) {
21     long long result = 0;
22     for (int i = l; i <= u; i++) {
23         result += array[i];
24     }
25     return result;
26 }
27
28 void get_new_array(int old_array[], int n, int new_array[]) {
29     int k = 0;
30     for (int i = 0; i < n; i++) {
31         for (int j = i; j < n; j++) {
32             new_array[k++] = (int)get_sum(old_array, i, j);
33         }
34     }
35 }
36
37 int main() {
38     int t;
39     scanf("%d", &t);
40     for (int i = 0; i < t; i++) {
41         printf("case #d:\n", i);
42         int n, m;
43         scanf("%d%d", &n, &m);
44         for (int j = 0; j < n; j++) {
45             scanf("%d", old_array + j);
46         }
47         get_new_array(old_array, n, new_array);
48         qsort(new_array, n * (n + 1) / 2, sizeof(int), cmp); // 新数组的大小不
是n
49         for (int j = 0; j < m; j++) {
50             int l, u;
51             scanf("%d%d", &l, &u);
52             // 输入数据从1开始, 存储从0开始
53             printf("%lld\n", get_sum(new_array, l - 1, u - 1));
54         }
55     }
56 }
57

```

./1036.数组相对排序.c

```

1 // 18:57
2 // 19:24
3
4 #include<stdio.h>
5 #include<stdlib.h>
6 #include<string.h>
7 #define N 500

```

```

8  #define M 1001
9
10 int a_map[M]; // 数组A的元素到位置的映射
11
12 int cmp(const void *pa, const void *pb) {
13     int a = *(int*)pa;
14     int b = *(int*)pb;
15     if (a_map[a] != a_map[b]) return a_map[a] - a_map[b];
16     return a - b;
17 }
18
19 int main() {
20     int b[N];
21     memset(a_map, 0x3f3f3f3f, M * sizeof(int)); // 正好前60个测试用例对了, 后40个
测试用例不对, 而前60个都是小数据量, 后40个都是大数据量, 原来是初始化的时候忘记乘
sizeof(int)了
22     int m, n;
23     scanf("%d", &m);
24     int temp;
25     for (int i = 0; i < m; i++) {
26         scanf("%d", &temp);
27         a_map[temp] = i;
28     }
29     scanf("%d", &n);
30     for (int i = 0; i < n; i++) {
31         scanf("%d", b + i);
32     }
33     qsort(b, n, sizeof(int), cmp);
34     for (int i = 0; i < n; i++) {
35         printf("%d ", b[i]);
36     }
37 }
38

```

./1037.一元多项式乘法.c

```

1  // 20:00
2  // 21:12
3
4  #include<stdio.h>
5  #include<string.h>
6  #include<ctype.h>
7  #define N 50
8
9  typedef int Polynomial_low[N];
10 typedef int Polynomial_high[2 * N];
11
12 char *read_one_term(char *s, Polynomial_low poly) {
13
14 }
15
16 void read_polynomial(char *s, Polynomial_low poly) {
17     memset(poly, 0, sizeof(Polynomial_low));
18     char *ps = s;

```

```

19 while (*ps) {
20     int sign = 1, coefficient = 1, degree = 1;
21     switch (*ps) { // 先符号
22         case '-':
23             sign = -1;
24         case '+':
25             ps++;
26             break;
27     }
28     if (isdigit(*ps)) { // 再系数
29         coefficient = 0;
30         while (isdigit(*ps)) {
31             coefficient = coefficient * 10 + *ps - '0';
32             ps++;
33         }
34     }
35     if (!*ps) { // 数字之后可能就读完了
36         degree = 0;
37     }
38     if (*ps == 'x') ps++;
39     if (*ps == '^') { // 次数
40         ps++;
41         degree = 0;
42         while (isdigit(*ps)) {
43             degree = degree * 10 + *ps - '0';
44             ps++;
45         }
46     }
47     poly[degree] = sign * coefficient;
48 }
49 }
50
51 void multiply_polynomial(Polynomial_low a, Polynomial_low b, Polynomial_high
result) {
52     memset(result, 0, sizeof(Polynomial_high));
53     for (int i = 0; i < N; i++)
54         for (int j = 0; j < N; j++) {
55             result[i + j] += a[i] * b[j];
56         }
57 }
58
59 void output_polynomial_coefficient(Polynomial_high poly) {
60     int i = 2 * N - 1;
61     // for (; i > 0 && poly[i] == 0; i--); // 跳过所有次数大于0, 系数为0的项
62     for (; i >= 0; i--) {
63         if (poly[i] == 0) continue;
64         printf("%d ", poly[i]);
65     }
66     printf("\n");
67 }
68
69 int main() {
70     Polynomial_low a, b;
71     Polynomial_high result;
72     char temp[101];

```

```

73     while (scanf("%s", temp) != EOF) {
74         read_polynomial(temp, a);
75         scanf("%s", temp);
76         read_polynomial(temp, b);
77         multiply_polynomial(a, b, result);
78         output_polynomial_coefficient(result);
79     }
80 }
81

```

./1038.排版.c

```

1 // 13:50
2 // 14:01
3
4 // 14:42
5 // 16:02
6
7 #include<stdio.h>
8 #include<string.h>
9 #define N 2000
10
11 // 返回读入单词数量
12 int read_str(char *origin, char **result) {
13     // 用sscanf应该也可以, 这里用了strtok
14     char **p = result;
15     *p++ = strtok(origin, " ");
16     while ((*p++ = strtok(NULL, " ")) != NULL) ;
17     // 如果最终返回结果为NULL后这个NULL已经读进去了, 而且p++了, 所以要-1
18     return p - result - 1;
19 }
20
21 void ouput_one_line(char **p, char **line_start, int temp_len, int m) {
22     if (p == line_start) {
23         printf("%s\n", *p);
24         return;
25     }
26     int extra_space_num = m - temp_len;
27     int min_space = extra_space_num / (p - line_start) + 1;
28     int big_space_num = extra_space_num % (p - line_start);
29     int small_space_num = p - line_start - big_space_num;
30     for (char **word = line_start; word < p; word++, small_space_num--) {
31         printf("%s", *word);
32         for (int i = 0; i < min_space; i++) putchar(' ');
33         if (small_space_num <= 0) putchar(' ');
34     }
35     printf("%s\n", *p);
36 }
37
38 void output_str(char **str, int words_num, int m) {
39     char **p = str, **end = str + words_num, **line_start = str;
40     int temp_len = strlen(*p); // 第一个单词
41     for (p++; p < end; p++) {
42         temp_len += strlen(*p) + 1; // 每个单词和左边至少一个空格

```

```

43     if (temp_len < m) continue;
44     if (temp_len > m) temp_len -= strlen(*p--) + 1;
45     output_one_line(p, line_start, temp_len, m);
46     p++;
47     line_start = p;
48     temp_len = strlen(*p); // 第一个单词左边无空格
49 }
50 // 此时p == end, 为NULL
51 // 下面这个情况不可能出现, 所以注释了
52 // if (temp_len > m) { // 这样也可能导致困惑, 但懒得改
53 //     temp_len -= strlen(*p--) + 1;
54 //     output_one_line(p, line_start, temp_len, m);
55 //     p++;
56 //     line_start = p;
57 //     temp_len = strlen(*p); // 第一个单词左边无空格
58 // }
59 if (temp_len > 0) {
60     output_one_line(p - 1, line_start, m, m); // 这样传参可能会导致歧义, 但懒
得改
61 }
62 }
63
64 int main() {
65     int t;
66     scanf("%d", &t);
67     for (int i = 0; i < t; i++) {
68         int m;
69         scanf("%d", &m);
70         char s[N + 2]; // 换行符和末尾'\0'
71         char *words[N + 1]; // NULL可能会存进去
72         getchar(); // 换行符
73         fgets(s, N+2, stdin);
74         s[strlen(s) - 1] = 0; // 换行符去掉
75         int n = read_str(s, words);
76         printf("case #d:\n", i);
77         output_str(words, n, m);
78     }
79 }
80

```

./1039.字符组合.c

```

1 // 16:08
2 // 16:51
3
4 #include<stdio.h>
5 #include<string.h>
6 #define N 16
7
8 // 返回读取的长度
9 int get_unrepeated_chars(char *origin, char *result) {
10     char *p_result = result;
11     int appear[128] = {0};
12     for (char *p = origin; *p; p++) appear[*p]++;

```

```

13     for (int i = 0; i < 128; i++) {
14         if (appear[i] > 0) *p_result++ = i;
15     }
16     *p_result = 0;
17     return p_result - result;
18 }
19
20 // 递归
21 void output_combinations(const char *prefix, char *s, int len) {
22     // 不用const会报错
23     if (len == 1) {
24         printf("%s", prefix);
25         putchar(*s);
26         putchar('\n');
27     }
28     if (len <= 1) return;
29     printf("%s", prefix);
30     putchar(*s);
31     putchar('\n'); // 只输出当前位
32     // putchar(*s);
33     char new_prefix[N + 1];
34     char appending[] = {*s, 0};
35     strcpy(new_prefix, prefix);
36     strcat(new_prefix, appending); // 不能把char类型的传给char*类型的
37     output_combinations(new_prefix, s + 1, len - 1); // 输出当前位
38     output_combinations(prefix, s + 1, len - 1); // 不输出当前位
39 }
40
41 int main() {
42     int t;
43     scanf("%d", &t);
44     for (int i = 0; i < t; i++) {
45         printf("case #d:\n", i);
46         char unrepeated[N + 1], origin[N + 1];
47         scanf("%s", origin);
48         int len = get_unrepeated_chars(origin, unrepeated);
49         output_combinations("", unrepeated, len);
50     }
51 }
52

```

./1040.字符串消除.c

```

1 // 18:40
2 // 19:59
3 // 20:28
4
5 #include<stdio.h>
6 #include<string.h>
7 #define N 100
8
9 // 返回消除掉的字符串个数
10 int eliminated_once(const char *origin, char *result) {
11     const char *end = origin + strlen(origin);

```



```

12     int eliminated_num = 0;
13     char *q = result;
14     for (const char *p = origin; p < end - 1; p++) {
15         if (*p != *(p + 1)) *q++ = *p;
16         else {
17             const char *r = p + 1;
18             while (r < end && *(r - 1) == *r) r++; // 找到不同的
19             eliminated_num += r - p;
20             p = r - 1; // 下一步到for循环里还要++的, 所以这里-1
21         }
22     }
23     if (*(end - 2) != *(end - 1)) *q++ = *(end - 1); // 最后一个字符
24     *q = 0; // 别忘了结束的0
25     return eliminated_num;
26 }
27
28 // 返回消除掉的字符串个数
29 int get_eliminated_string(const char *origin, char *result) {
30     char temp[N + 1];
31     strcpy(temp, origin);
32     int eliminated_num = 0;
33     int once = 0;
34     while ((once = eliminated_once(temp, result)) > 0) {
35         eliminated_num += once;
36         strcpy(temp, result);
37     }
38     return eliminated_num;
39 }
40
41 // 得到字符串中某个位置的回文子串长度
42 int get_palindrome_num(const char *str, const char *p) {
43     int half = 1; // 朝向一边的偏移量, 算上自身
44     while (p - half >= str && *(p + half) &&
45           *(p - half) == *(p + half)) half++;
46     return 2 * half - 1;
47 }
48
49 // 得到最大回文子串长度, 用这个方法会有情况不对, 暂时还未发现原因
50 // BBABCABCCBCABCCBAABBBBACAAACBCBACCACAAACACBABBAAA
51 // 正确答案是44, 但用这样的方法的结果是43
52 // CACCACAABAABCCCAABCAAAAACCAABCBBABCCBABBCCBACCACACCCBBBAAAB
53 // 正确答案是58, 但用这样的方法结果是57
54 int get_max_palindrome_num(const char *str) {
55     int max_num = 0;
56     for (const char *p = str; *p; p++) {
57         int temp;
58         if ((temp = get_palindrome_num(str, p)) > max_num)
59             max_num = temp;
60     }
61     return max_num;
62 }
63
64 // pos是src中的某个字符的地址, 可以是末尾0的地址, 把c插入到pos前方, 可以返回插入后的长度但懒得返回
65 void insert_char(char *dst, const char *src, char c, const char *pos) {

```

```

66     for (; *src || *(src - 1); dst++) {
67         if (src == pos) *dst = c, pos = NULL; // 取消pos防止下次循环再运行到这里
68         else *dst = *src++;
69     }
70     *dst = 0;
71 }
72
73 int get_max_eliminated_num(const char *str) {
74     int max_num = 0;
75     for (const char *p = str; *p || *(p - 1); p++) { // 末尾的*p == 0就代表插
入到结尾, 不过这里懒得考虑空字符串的情况
76         char temp[N + 2], temp2[N + 2]; // 插入了一个字符
77         for (char c = 'A'; c <= 'C'; c++) {
78             insert_char(temp, str, c, p);
79             int num = get_eliminated_string(temp, temp2);
80             if (num > max_num) max_num = num;
81         }
82     }
83     return max_num;
84 }
85
86 int main() {
87     int t;
88     char s[N + 1];
89     scanf("%d", &t);
90     for (int i = 0; i < t; i++) {
91         scanf("%s", s);
92         printf("case #%d:\n", i);
93         // char eliminated_string[N + 1];
94         // int eliminated_num = get_eliminated_string(s,
eliminated_string);
95         // int max_substring_len =
get_max_palindrome_num(eliminated_string);
96         // printf("%d\n", eliminated_num + max_substring_len + 1);
97         printf("%d\n", get_max_eliminated_num(s));
98     }
99 }
100

```

./1041.十六进制.c

```

1 // 20:31
2 // 20:48
3 // 20:59
4
5 #include<stdio.h>
6 #include<stdlib.h>
7 #include<string.h>
8 #define N 100000
9
10 int main() {
11     // printf("%lld\n", atoll("0x10")); // 失败
12     // printf("%lld\n", strtoull("0x10", NULL, 16)); // 成功
13     // printf("%lld\n", strtoull("10", NULL, 16)); // 成功

```

```

14 // printf("%lld\n", strtoull("0x10g8", NULL, 16)); // 成功
15 // printf("%lld\n", strtoull("ggggg", NULL, 16)); // 结果是0
16 char s[N + 1];
17 scanf("%s", s);
18 char *p = s;
19 int has_num = 0;
20 while ((p = strstr(p, "0x")) != NULL) {
21     // 可能出现0xggggggggx000x的情况, 提取出的p转成数字是0, 但实际上不应该算上这种
    情况
22     if (strlen(p) > 2 &&
23         (*(p + 2) >= '0' && *(p + 2) <= '9') ||
24         (*(p + 2) >= 'a' && *(p + 2) <= 'f')) {
25         printf("%llu ", strtoull(p, NULL, 16));
26         has_num = 1;
27     }
28     p += 2;
29 }
30 if (!has_num) printf("-1");
31 }
32

```

./1042.字符串变换.c

```

1 // 8:05
2 // 9:52
3 // 10:10
4 // 10:50
5
6 #include<stdio.h>
7 #include<stdlib.h>
8 #define N 100
9 #define M 100000
10
11 typedef struct {
12     char c;
13     int times;
14 } str[N];
15 typedef long long ll;
16 str list[M];
17
18 int read_str(const char *s, str dst) {
19     int i = 0;
20     for (const char *p = s; *p; p++) {
21         const char *q = p;
22         for (q++; *q && *p == *q; q++);
23         dst[i].c = *p;
24         dst[i].times = q - p;
25         i++;
26         p = q - 1; // 之后一步要p++
27     }
28     return i;
29 }
30
31 // 共n行, 每行m个不连续相同字符

```

```

32 int is_possible(int n, int m) {
33     for (int i = 1; i < n; i++) {
34         for (int j = 0; j < m; j++) {
35             if (list[i][j].c != list[0][j].c) return 0;
36         }
37     }
38     return 1;
39 }
40
41 // 其中一个字符全部变换到target次数所需要的最小次数
42 ll get_one_col_min_times(int n, int j, int target) {
43     ll result = 0;
44     for (int i = 0; i < n; i++) {
45         result += abs(list[i][j].times - target);
46     }
47     return result;
48 }
49
50 int cmp(const void *pa, const void *pb) {
51     int a = *(int*)pa;
52     int b = *(int*)pb;
53     return a - b;
54 }
55
56 // 无法达到目标时返回-1
57 ll get_min_change_times(int n, int m) {
58     if (!is_possible(n, m)) return -1LL;
59     ll result = 0;
60     for (int j = 0; j < m; j++) {
61         // 不是距离平均数的和最小, 而是中位数
62         int times[M];
63         for (int i = 0; i < n; i++) times[i] = list[i][j].times;
64         qsort(times, n, sizeof(int), cmp);
65         // for (int i = 0; i < n; i++) { // 这样复杂度太高了, 超时
66         //     int less_than_medium = 0;
67         //     medium = list[i][j].times;
68         //     for (int k = 0; k < n; k++)
69         //         if (list[k][j].times < medium) less_than_medium++;
70         //     if (less_than_medium == n / 2) break;
71         //     // avg += (double)list[i][j].times / n;
72         // }
73         int medium = times[n / 2];
74         ll temp1 = get_one_col_min_times(n, j, medium);
75         ll temp2 = get_one_col_min_times(n, j, medium + 1);
76         result += temp1 < temp2 ? temp1 : temp2;
77     }
78     return result;
79 }
80
81 ll read_and_get_times(int n) {
82     int m = -1;
83     for (int i = 0; i < n; i++) {
84         char s[N + 1];
85         scanf("%s", s);
86         int temp = read_str(s, list[i]);

```

```

87     if (m < 0) m = temp;
88     else if (m != temp) return -1LL; // 提前排除掉不重复字符数量不同的情况
89     }
90     return get_min_change_times(n, m);
91 }
92
93 int main() {
94     int n;
95     scanf("%d", &n);
96     printf("%lld", read_and_get_times(n));
97 }
98

```

./1054.负二进制.c

```

1 // 10:00
2 // 11:00
3 // 14:50
4 // 15:01
5 // 15:06
6
7 #include<stdio.h>
8 #include<string.h>
9 #define N 100
10
11 typedef struct {
12     int nums[N]; // 从低位到高位
13     int count;
14     int sign;
15 } BigInt;
16
17 void str_to_BigInt(char *s, BigInt *bigint) {
18     int len = strlen(s);
19     for (int i = len - 1; i >= 1; i--) {
20         bigint->nums[bigint->count++] = s[i] - '0';
21     }
22     if (s[0] == '-')
23         bigint->sign = -1;
24     else
25         bigint->nums[bigint->count++] = s[0] - '0';
26 }
27
28 void carry(BigInt *bigint) {
29     int i = 0;
30     for (; i < N; i++) {
31         if (bigint->nums[i] >= 10) {
32             bigint->nums[i + 1] += bigint->nums[i] / 10;
33             bigint->nums[i] %= 10;
34         }
35     }
36     for (i = N - 1; i >= 0; i--) {
37         if (bigint->nums[i]) break;
38     }
39     bigint->count = i + 1;

```

```

40 }
41
42 void add1(BigInt *bigint) {
43     bigint->nums[0] ++;
44     carry(bigint);
45 }
46
47 // 返回余数
48 int div_2(BigInt *bigint) {
49     int temp = 0;
50     int i;
51     for (i = bigint->count - 1; i > 0; i--) {
52         temp = temp * 10 + bigint->nums[i];
53         if (temp < 2) {
54             bigint->nums[i] = 0;
55             temp = temp * 10 + bigint->nums[--i];
56         }
57         bigint->nums[i] = temp / 2;
58         temp %= 2;
59     }
60     if (i >= 0) {
61         temp = temp * 10 + bigint->nums[0];
62         bigint->nums[0] = temp / 2;
63     }
64     carry(bigint);
65     return temp % 2;
66 }
67
68 // 返回余数
69 int div_neg_2(BigInt *bigint) {
70     int remainder = div_2(bigint);
71     if (bigint->sign < 0 && remainder == 1) {
72         // remainder += 2;
73         add1(bigint);
74     }
75     bigint->sign = - bigint->sign;
76     return remainder;
77 }
78
79 void print_neg_2_base(BigInt *bigint) {
80     char temp[N * 4];
81     int i = 0;
82     if (bigint->count <= 1 && bigint->nums[0] == 0) {
83         putchar('0');
84         return;
85     }
86     while (bigint->count > 1 || bigint->nums[0] > 0) {
87         temp[i++] = div_neg_2(bigint) + '0';
88     }
89     for (i--; i >= 0; i--) {
90         printf("%c", temp[i]);
91     }
92 }
93
94 int main() {

```

```

95     BigInt n = {{0}, 0, 1};
96     char s[N + 10];
97     scanf("%s", s);
98     str_to_BigInt(s, &n);
99     print_neg_2_base(&n);
100 }
101

```

./1055.数字排序.c

```

1 // 8:02
2 // 8:29
3
4 #include<stdio.h>
5 #include<stdlib.h>
6
7 typedef struct {
8     double value;
9     char str[101];
10 } num;
11
12 num nums[100];
13
14 int cmp(const void *pa, const void *pb) {
15     num a = *(num*)pa;
16     num b = *(num*)pb;
17     if (a.value == b.value) {
18         return 0;
19     }
20     if (a.value > b.value) {
21         return 1;
22     } else {
23         return -1;
24     }
25 }
26
27 int main() {
28     int n;
29     char str[101];
30     scanf("%d", &n);
31     for (int i = 0; i < n; i++) {
32         scanf("%s", nums[i].str);
33         nums[i].value = atof(nums[i].str);
34     }
35     qsort(nums, n, sizeof(num), cmp);
36     for (int i = 0; i < n; i++) {
37         printf("%s\n", nums[i].str);
38     }
39 }
40

```

./1055.计算多项式的系数.c

```

1 // 10:06
2 // 11:50
3 // 16:10
4 // 16:32
5 #include<stdio.h>
6 #define MOD_NUM 10007
7
8 /* 计算a*b对MOD_NUM取模的值 */
9 int multiply(int a, int b) {
10     long long temp = a % MOD_NUM;
11     temp *= b;
12     int result = temp % MOD_NUM;
13     return result;
14 }
15
16 /* 计算a的n次方对MOD_NUM取模的值 */
17 int power(int a, int n) {
18     int result = 1;
19     for (; n > 0; n--) { // 循环n次
20         result = multiply(result, a);
21     }
22     return result;
23 }
24
25 /* 计算C(k, n)的组合数对MOD_NUM取模的值 */
26 //int combination(int k, int n) {
27 //    int i = .
28 //    int ai = 1; // 每一项乘起来
29 //    int ai = multiply(1, k); // 从k乘到k-n+1, 共n个
30 //    int result = 1;
31 //    int denominator = multiply(1, n); // 从n乘到1
32 //    for (int i = 1; i <= n; i++) { // 循环n-1次
33 //        int remainder = k % i;
34 //        int ai = k - remainder;
35 //        if (remainder == 0) {
36 //            ai /= n;
37 //        } else {
38 //            ai /= remainder;
39 //        }
40 //        result = multiply(result, ai);
41 //        denominator = multiply(denominator, n);
42 //    }
43 //    int result = numerator / denominator;
44 //    if (result == 0) {
45 //        result = numerator;
46 //    }
47 //    return result;
48 //}
49
50 int combinations[1001][1001] = {{0}};
51 void init_combination_triangle() {
52     combinations[0][0] = 1;
53     for (int i = 1; i < 1001; i++) {
54         combinations[i][0] = 1;
55         for (int j = 1; j < i; j++) {

```



```

56         combinations[i][j] = combinations[i-1][j-1] + combinations[i-1]
[j];
57         combinations[i][j] %= MOD_NUM;
58     }
59     combinations[i][i] = 1;
60 }
61 }
62
63 int main() {
64     int t;
65     int a, b, k, n, m;
66     init_combination_triangle();
67     scanf("%d", &t);
68     for (int i = 0; i < t; i++) {
69         scanf("%d%d%d%d", &a, &b, &k, &n, &m);
70         int an = power(a, n);
71         int bm = power(b, m);
72         int comb_k_n = combinations[k][n];
73         int result = multiply(an, bm);
74         result = multiply(result, comb_k_n);
75         printf("case #d:\n%d\n", i, result);
76     }
77     return 0;
78 }
79
80

```

./1056.找出最小字符串.c

```

1 // 9:10
2
3 #include<stdio.h>
4 #define N 100
5
6 typedef struct {
7     int alpha;
8     int times;
9 } Repeat;
10 Repeat alphas[N];
11
12 int main() {
13     int c;
14     int i = 0;
15     while ((c = getchar()) != '\n') {
16         alphas[i].alpha = c;
17         alphas[i].times = 1;
18         i++;
19     }
20     for (int j = 0; j < i - 1; j++) {
21         int k = j + 1;
22         while (alphas[j].alpha == alphas[k].alpha)
23             k++;
24         if (alphas[j].alpha < alphas[k].alpha) {
25             for (int i = j; i < k; i++) {

```

```

26         alphas[i].times = 2;
27     }
28 }
29 }
30 for (int j = 0; j < i; j++) {
31     for (int k = 0; k < alphas[j].times; k++) {
32         putchar(alphas[j].alpha);
33     }
34 }
35 }
36
37

```

./1056.浮点数加法.c

```

1 // 19:06
2 // 19:09
3 // 19:36
4 // 09:07
5 // 09:53
6 // 10:17
7
8 #include<stdio.h>
9 #include<string.h>
10 #define MAX_LEN 510
11
12 typedef struct {
13     int only_nums[MAX_LEN * 2]; // 前半部分整数, 后半部分小数
14     int point_pos_from_little;
15 } Number;
16
17 void read_number(char str[], int len, Number *num) {
18     int i = 0;
19     int temp[MAX_LEN];
20     for (; i < len; i++) {
21         if (str[i] == '.') {
22             num->point_pos_from_little = len - 1 - i;
23             break;
24         }
25         temp[i] = str[i] - '0';
26     }
27     temp[i] = 0; // 此时i是temp的长度
28     memset(num->only_nums, 0, (MAX_LEN-i)*sizeof(int)); // 整数前面都是0
29     memcpy(&num->only_nums[MAX_LEN-i], temp, i*sizeof(int)); // 放到整数里
30     i++; // 跳过这个小数点
31     for (int j = 0; i < len; i++, j++) {
32         num->only_nums[MAX_LEN + j] = str[i] - '0';
33     }
34     memset(num->only_nums+MAX_LEN+i, 0, (MAX_LEN-i)*sizeof(int));
35 }
36
37 void carry(Number *num) {
38     for (int i = MAX_LEN * 2 - 1; i >= 0; i--) {
39         if (num->only_nums[i] >= 10) {

```

```

40         num->only_nums[i-1] += num->only_nums[i] / 10;
41         num->only_nums[i] %= 10;
42     }
43 }
44 }
45
46 void add(Number *a, Number *b, Number *result) {
47     memset(result->only_nums, 0, MAX_LEN*2*sizeof(int));
48     for (int i = MAX_LEN*2 - 1; i>=0; i--) {
49         result->only_nums[i] += a->only_nums[i] + b->only_nums[i];
50     }
51     carry(result);
52 }
53
54 void print_number(Number *num, int n) {
55     if (num->only_nums[MAX_LEN+n] >= 5) {
56         num->only_nums[MAX_LEN+n-1] += 1;
57         carry(num);
58     }
59     int i = 0;
60     while(i < MAX_LEN && num->only_nums[i] == 0) i++;
61     if (i >= MAX_LEN) {
62         putchar('0');
63     } else {
64         while (i < MAX_LEN) {
65             putchar(num->only_nums[i] + '0');
66             i++;
67         }
68     }
69     putchar('.');
70     for (int j = 0; j < n; j++, i++) {
71         putchar(num->only_nums[i] + '0');
72     }
73 }
74
75 int main() {
76     Number a={{0},0}, b={{0},0}, result={{0},0};
77     char str_a[MAX_LEN], str_b[MAX_LEN];
78     int n;
79     scanf("%s %s %d", str_a, str_b, &n);
80     read_number(str_a, strlen(str_a), &a);
81     read_number(str_b, strlen(str_b), &b);
82     add(&a, &b, &result);
83     print_number(&result, n);
84 }
85
86

```

./1057.整数分解.c

```

1 // 9:20
2 // 10:29
3
4 #include<stdio.h>

```

```

5 #define PRIME_NUM 168
6 #define MAX_N 1000
7
8 int primes[PRIME_NUM] =
  {2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,89,97,101,
  103,107,109,113,127,131,137,139,149,151,157,163,167,173,179,181,191,193,197,
  199,211,223,227,229,233,239,241,251,257,263,269,271,277,281,283,293,307,311,
  313,317,331,337,347,349,353,359,367,373,379,383,389,397,401,409,419,421,431,
  433,439,443,449,457,461,463,467,479,487,491,499,503,509,521,523,541,547,557,
  563,569,571,577,587,593,599,601,607,613,617,619,631,641,643,647,653,659,661,
  673,677,683,691,701,709,719,727,733,739,743,751,757,761,769,773,787,797,809,
  811,821,823,827,829,839,853,857,859,863,877,881,883,887,907,911,919,929,937,
  941,947,953,967,971,977,983,991,997};
9 long long plans[MAX_N][PRIME_NUM] = {{0}};
10
11 // n在只能用从0到prime_num个数字时的分解方法
12 long long get_one_plan(int n, int prime_num) {
13     long long result = plans[n][prime_num - 1];
14     // for (int i = 0; i < PRIME_NUM; i++) {
15     //     if (primes[i] == n) {
16     //         result ++;
17     //         break;
18     //     }
19     // }
20     while (n >= primes[prime_num]) {
21         n -= primes[prime_num];
22         result += plans[n][prime_num - 1];
23     }
24     if (n == 0) {
25         result ++;
26     }
27     return result;
28 }
29
30 void get_plans() {
31     for (int j = 0; j < PRIME_NUM; j++) {
32         // plans[0][j] = 0;
33         plans[2][j] = 1;
34         plans[3][j] = 1;
35     }
36     for (int i = 3; i < MAX_N; i++) {
37         if (i % 2 == 0) {
38             plans[i][0] = 1;
39         } else {
40             plans[i][0] = 0;
41         }
42     }
43     for (int i = 4; i < MAX_N; i++) {
44         for (int j = 1; j < PRIME_NUM; j++) {
45             plans[i][j] = get_one_plan(i, j);
46         }
47     }
48 }
49
50 int main() {

```

```

51     int n;
52     get_plans();
53     scanf("%d", &n);
54     printf("%lld", plans[n][PRIME_NUM - 1]);
55 }
56

```

./1057.计算a的n次方的大整数.c

```

1 // 20:11
2 // 20:17
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <string.h>
6 #define MAX 200
7
8 typedef struct
9 {
10     int len;
11     int s[MAX+1];
12 } hp;
13
14 void input(hp *a, int x) //读入数字
15 {
16     int i;
17
18     a->len = 0;
19
20     while (x > 0)
21     {
22         a->s[1 + a->len++] = x % 10;
23         x /= 10;
24     }
25
26     for (i = a->len + 1; i <= MAX; i++)
27         a->s[i] = 0;
28 }
29
30 void print(hp *y) //打印数字
31 {
32     int i;
33     for (i = y->len; i >= 1; i--)
34         printf("%d", y->s[i]);
35     printf("\n");
36 }
37
38 void multiply(hp *a, hp *b, hp *c) //高精度 * 高精度
39 {
40     int i, j, len;
41
42     for (i = 1; i <= MAX; i++) c->s[i] = 0;
43
44     for (i = 1; i <= a->len; i++)
45     {

```

```

46     for (j = 1; j <= b->len; j++)
47     {
48         c->s[i+j-1] += a->s[i] * b->s[j];
49         c->s[i+j] += c->s[i+j-1] / 10;
50         c->s[i+j-1] %= 10;
51     }
52 }
53
54 len = a->len + b->len + 1;
55 while (len > 1 && c->s[len] == 0) len--;
56 c->len = len;
57 }
58
59 void power(hp *a, int b, hp *c) //高精度乘方  $c = a^b$ 
60 {
61     hp e;
62
63     if (b == 0)
64     {
65         c->len = 1;
66         c->s[1] = 1;
67     }
68     else if (b == 1)
69     {
70         memcpy(c, a, sizeof(hp));
71     }
72     else
73     {
74         power(a, b / 2, &e);
75         multiplyh(&e, &e, c);
76
77         if (b % 2 == 1)
78         {
79             memcpy(&e, c, sizeof(hp));
80             multiplyh(&e, a, c);
81         }
82     }
83 }
84
85 int main() {
86     int t;
87     scanf("%d", &t);
88     int a, n;
89     for (int i = 0; i < t; i++) {
90         scanf("%d%d", &a, &n);
91         hp a_hp, result;
92         input(&a_hp, a);
93         power(&a_hp, n, &result);
94         printf("case #%d:\n", i);
95         print(&result);
96     }
97     return 0;
98 }
99

```

./1058.一个游戏.c

```
1 // 20:18
2 // 20:27
3 // 20:36
4
5 #include<stdio.h>
6 #include<string.h>
7
8 int main() {
9     int t;
10    scanf("%d", &t);
11    for (int i = 0; i < t; i++) {
12        int n;
13        int head[100];
14        int tail[100];
15        int a_to_b[100][100];
16        scanf("%d", &n);
17        memset(head, 0, 100*sizeof(int));
18        memset(tail, 0, 100*sizeof(int));
19        memset(a_to_b, 0, 100*100*sizeof(int));
20        int justify = 1;
21        for (int j = 0; j < n; j++) {
22            int ai, bi;
23            scanf("%d%d", &ai, &bi);
24            head[ai] ++;
25            tail[bi] ++;
26            if (a_to_b[ai][bi]) { // 去重
27                continue;
28            } else {
29                a_to_b[ai][bi] = 1;
30            }
31            if (head[ai] > 1 || head[ai] > 0 && tail[ai] > 0
32                || head[bi] > 0 && tail[bi] > 0) {
33                justify = 0;
34            }
35        }
36        if (justify) {
37            printf("Lucky dxw!\n");
38        } else {
39            printf("Poor dxw!\n");
40        }
41    }
42 }
43
```

./1059.计算a的n次方的大整数.c

```
1 // 10:14
2 // 10:49
3
4 #include<stdio.h>
```

```

5  #include<string.h>
6  #define N 1005
7
8  typedef struct {
9      int num[N]; // 从低位到高位
10     int cnt; // 有效数字位数
11 } BigInt;
12
13 void init_num(BigInt *num) {
14     memset(num->num, 0, N * sizeof(int));
15     num->cnt = 0;
16 }
17
18 void read_num(int a, BigInt *result) {
19     init_num(result);
20     result->num[0] = a;
21     result->cnt = 1;
22 }
23
24 void output_num(BigInt *num) {
25     for (int i = num->cnt - 1; i >= 0; i--) {
26         putchar(num->num[i] + '0');
27     }
28     putchar('\n');
29 }
30
31 void carry(BigInt *num) {
32     int i = 0;
33     for (; i < num->cnt; i++) {
34         if (num->num[i] >= 10) {
35             num->num[i + 1] += num->num[i] / 10;
36             num->num[i] %= 10;
37         }
38     }
39     for (; i < N; i++) {
40         if (num->num[i] > 0) {
41             num->cnt = i + 1;
42         }
43         if (num->num[i] >= 10) {
44             num->num[i + 1] += num->num[i] / 10;
45             num->num[i] %= 10;
46         }
47     }
48 }
49
50 void cpy_num(BigInt *dst, BigInt *src) {
51     memcpy(dst, src, sizeof(BigInt));
52 }
53
54 void multiply(BigInt *a, BigInt *b, BigInt *result) {
55     init_num(result);
56     // int weight = 1;
57     for (int i = 0; i < a->cnt; i++) {
58         for (int j = 0; j < b->cnt; j++) {
59             result->num[i + j] += a->num[i] * b->num[j];

```



```

60     }
61     // weight *= 10;
62 }
63 carry(result);
64 }
65
66 void pow(BigInt *a, int n, BigInt *result) {
67     // init_num(result);
68     BigInt temp;
69     read_num(1, result);
70     for (int i = 0; i < n; i++) {
71         multiply(a, result, &temp);
72         cpy_num(result, &temp);
73     }
74 }
75
76 int main() {
77     int t;
78     int a, n;
79     BigInt big_a, result;
80     scanf("%d", &t);
81     for (int i = 0; i < t; i++) {
82         printf("case #%d:\n", i);
83         scanf("%d%d", &a, &n);
84         read_num(a, &big_a);
85         pow(&big_a, n, &result);
86         output_num(&result);
87     }
88 }
89

```

./1061.计算n! 右端0的个数(II).c

```

1 // 8:06
2 // 8:23
3 // 8:44
4
5 #include<stdio.h>
6
7 int get_5_factor_nums_in_factorial(int num) {
8     int result = 0;
9     for (int i = 1; i <= num; i++) {
10         int temp = i;
11         while (temp % 5 == 0) {
12             temp /= 5;
13             result++;
14         }
15     }
16     return result;
17 }
18
19 int main() {
20     int t;
21     scanf("%d", &t);

```

```

22     for (int i = 0; i < t; i++) {
23         int n;
24         scanf("%d", &n);
25         printf("case #d:\n", i);
26         printf("%d\n", get_5_factor_nums_in_factorial(n));
27     }
28 }
29

```

./1062.计算2的N次方.c

```

1 // 8:59
2 // 9:02
3
4 #include<stdio.h>
5
6 int main() {
7     int t;
8     scanf("%d", &t);
9     for (int i = 0; i < t; i++) {
10        int n;
11        scanf("%d", &n);
12        printf("case #d:\n%d\n", i, 1<<n);
13    }
14 }
15

```

./1063.二进制倒置.c

```

1 // 9:04
2 // 9:15
3 // 9:36
4 // 10:01
5
6 #include<stdio.h>
7 #include<string.h>
8 #define N 101
9 #define M 334
10
11 // 倒置
12 typedef struct {
13     int s[N];
14     int cnt;
15 } Big;
16
17 void init_big(Big *num) {
18     memset(num, 0, sizeof(Big));
19     // num->cnt = 1; // 初始化为0的时候是0位数字
20 }
21
22 void read(char s[], Big *num) {
23     init_big(num);

```

```

24     for (int i = strlen(s) - 1; i >= 0; i--) {
25         num->s[num->cnt++] = s[i] - '0';
26     }
27 }
28
29 void output(Big *num) {
30     for (int *p = num->s + num->cnt - 1; p >= num->s; p--) {
31         putchar(*p + '0');
32     }
33     putchar('\n');
34 }
35
36 // 顺便更新num->cnt
37 void carry(Big *num) {
38     for (int i = 0; i < N - 1; i++) {
39         if (num->s[i] >= 10) {
40             num->s[i + 1] += num->s[i] / 10;
41             num->s[i] %= 10;
42         }
43     }
44     int i = N - 1;
45     for (; i > 0; i--) {
46         if (num->s[i] != 0) break;;
47     }
48     num->cnt = i + 1; // 如果数字为0, cnt是1
49 }
50
51 // 返回余数
52 int divide_by_2(Big *num) {
53     int borrow = 0;
54     for (int i = num->cnt - 1; i > 0; i--) {
55         borrow += num->s[i];
56         if (borrow < 2){
57             num->s[i] = 0;
58             borrow *= 10;
59             continue;
60         }
61         num->s[i] = borrow >> 1;
62         borrow &= 1;
63         borrow *= 10;
64     }
65     borrow += num->s[0];
66     num->s[0] = borrow >> 1;
67     borrow &= 1;
68     carry(num);
69     return borrow;
70 }
71
72 // 返回二进制位数
73 int read_to_binary(Big *num, int binary[]) {
74     int *p = binary;
75     for (; num->cnt != 1 || num->s[0] != 0; p++) {
76         *p = divide_by_2(num);
77     }
78     return p - binary;

```

```

79 }
80
81 void multiply_2(Big *num) {
82     for (int i = 0; i < num->cnt; i++) num->s[i] <<= 1;
83     carry(num);
84 }
85
86 void add_1(Big *num) {
87     num->s[0] += 1;
88     carry(num);
89 }
90
91 void binary_to_big(int binary[], int binary_len, Big *num) {
92     init_big(num);
93     num->cnt = 1; // 初始化后作为1处理
94     for (int i = 0; i < binary_len - 1; i++) {
95         if (binary[i]) add_1(num);
96         multiply_2(num);
97     }
98     if (binary[binary_len - 1]) add_1(num);
99 }
100
101 int main() {
102     int t;
103     scanf("%d", &t);
104     for (int i = 0; i < t; i++) {
105         char s[N + 1];
106         scanf("%s", &s);
107         Big n;
108         read(s, &n);
109         int binary[M];
110         int binary_len = read_to_binary(&n, binary);
111         binary_to_big(binary, binary_len, &n);
112         printf("case #%d:\n", i);
113         output(&n);
114     }
115 }
116

```

./1064.A-B(Big Integer).c

```

1 // 10:11
2
3 #include<stdio.h>
4 #include<string.h>
5 #define N 500
6
7 // 倒置
8 typedef struct {
9     int s[N];
10    int cnt;
11    int sign;
12 } Big;
13

```

```

14 void init(Big *num) {
15     memset(num, 0, sizeof(Big));
16     num->sign = 1;
17 }
18
19 void input(Big *num, char s[]) {
20     init(num);
21     for (char *p = s + strlen(s) - 1; p >= s; p--) {
22         num->s[num->cnt++] = *p - '0';
23     }
24 }
25
26 void output(Big *num) {
27     if (num->sign < 0) putchar('-');
28     for (int i = num->cnt - 1; i > 0; i--) {
29         putchar(num->s[i]);
30     }
31     putchar('\n');
32 }
33
34 void carry(Big *num) {
35     for (int i = 0; i < N - 1; i++) {
36         if (num->s[i] >= 10) {
37             num->s[i + 1] = num->s[i] / 10;
38             num->s[i] /= 10;
39         }
40     }
41     int i = N - 1;
42     for (; i > 0; i--) {
43         if (num->s[i] != 0) break;
44     }
45     num->cnt = i;
46 }
47
48 int cmp(Big *a, Big *b) {
49     if (a->cnt != b->cnt) return a->cnt - b->cnt;
50     for (int i = a->cnt - 1; i >= 0; i--) {
51         if (a->s[i] == b->s[i]) continue;
52         if (a->s[i] > b->s[i]) return 1;
53         else return -1;
54     }
55     return 0;
56 }
57
58 void minus(Big *a, Big *b, Big *result) {
59     init(result);
60     int borrow = 0;
61     for (int i = N - 1; i >= 0; i++) {
62         borrow += a->s[i];
63         if (borrow < b->s[i]) {
64             result->s[i + 1]--;
65             borrow += 10;
66         }
67         result->s[i] = borrow - b->s[i];
68         borrow = 0;

```

```

69     }
70     carry(result);
71 }
72
73 int main() {
74     char a[N + 1], b[N + 1];
75     Big A, B, result;
76     while (scanf("%s", a) != EOF) {
77         scanf("%s", b);
78         input(&A, a);
79         input(&B, b);
80         if (cmp(&A, &B) < 0) {
81             minus(&B, &A, &result);
82             result.sign = -1;
83         }
84         minus(&A, &B, &result);
85         output(&result);
86     }
87 }
88

```

./1065.浮点数减法.c

```

1 // 10:34
2 // 11:55
3
4 // 13:32
5 // 14:53
6
7 #include<stdio.h>
8 #include<string.h>
9 #define N 600
10 #define M 50
11
12 typedef struct {
13     int s[2 * N];
14     int cnt; // 最右边的非零数字的位置
15     int sign;
16 } BigFloat;
17 // 前N位是小数, 后N位是整数
18
19 void init(BigFloat *num) {
20     memset(num, 0, sizeof(BigFloat));
21     num->sign = 1;
22     // num->cnt = 1;
23 }
24
25 void read_num(BigFloat *num, char *s) {
26     init(num);
27     if (*s == '-') {
28         num->sign = -1;
29         s++;
30     }
31     char *p_point = strchr(s, '.');

```

```

32     char *end = s + strlen(s);
33     int *p_result = num->s + N;
34     if (p_point == NULL) p_point = end;
35     for (char *p = p_point + 1; p < end; p++) {
36         *--p_result = *p - '0';
37     }
38     p_result = num->s + N;
39     for (char *p = p_point - 1; p >= s; p--) {
40         *p_result++ = *p - '0';
41     }
42     //     num->cnt = p - num->s - N;
43 }
44
45 int cmp(BigFloat *a, BigFloat *b) {
46     for (int i = 2 * N - 1; i >= 0; i--) {
47         if (a->s[i] == b->s[i]) continue;
48         return a->s[i] - b->s[i];
49     }
50     return 0;
51 }
52
53 void carry(BigFloat *num, int n) {
54     for (num->cnt = 2 * N - 1; num->cnt > 0; num->cnt--) {
55         if (num->s[num->cnt] != 0) break;
56     }
57     for (int i = 0; i <= num->cnt; i++) {
58         if (num->s[i] < 0) {
59             num->s[i] += 10;
60             num->s[i + 1] --;
61         }
62         if (num->s[i] >= 10) {
63             num->s[i + 1] += num->s[i] / 10;
64             num->s[i] %= 10;
65         }
66     }
67     for (num->cnt = 2 * N - 1; num->cnt > 0; num->cnt--) {
68         if (num->s[num->cnt] != 0) break;
69     }
70     if (num->s[num->cnt] < 0) {
71         for (int i = 0; i < 2 * N; i++)
72             num->s[i] *= -1;
73         num->sign = -1;
74         carry(num, n);
75     }
76 }
77
78 void minus(BigFloat *a, BigFloat *b, BigFloat *result, int n) {
79     init(result);
80     for (int i = 0; i < 2 * N; i++) {
81         result->s[i] = a->s[i] - b->s[i];
82     }
83     //     if (result->s[N - n - 1] >= 5) result->s[N - n]++;
84     carry(result, n);
85     if (result->s[N - n - 1] >= 5) {
86         result->s[N - n]++;

```

```

87     carry(result, n);
88     }
89 }
90
91 //void minus_with_cmp(BigFloat *a, BigFloat *b, BigFloat *result) {
92 //
93 //}
94
95 void output(BigFloat *num, int n){
96     int *p = num->s;
97     if (num->sign < 0) {
98         printf("-");
99         p++;
100    }
101    int i = 2 * N - 1;
102    for (; i > N; i--) {
103        if (num->s[i] != 0) break;
104    }
105    for (; i >= N; i--) {
106        putchar(num->s[i] + '0');
107    }
108    putchar('.');
109    // int end_0 = 0;
110    // for (; end_0 < N; end_0++) {
111    //     if (num->s[end_0]) break;
112    // }
113    for (int i = N - 1; i >= N - n; i--) {
114        putchar(num->s[i] + '0');
115    }
116    putchar('\n');
117 }
118
119 int main() {
120     char s[N + 1];
121     BigFloat a, b, result;
122     scanf("%s", s);
123     read_num(&a, s);
124     scanf("%s", s);
125     read_num(&b, s);
126     int n;
127     scanf("%d", &n);
128     minus(&a, &b, &result, n);
129     output(&result, n);
130     return 0;
131 }
132
133

```

./1081.种田.c

```

1 // 9:28
2 // 9:44
3
4 #include<stdio.h>

```



```

5
6 // long long body_power[;]
7
8 long long get_body_power(long long x, long long y) {
9     long long max_square_num = y / x;
10    long long result = 4 * x * max_square_num;
11    y %= x;
12    if (y <= 0)
13        return result;
14    // if (y > x)
15    //     return result + get_body_power(x, y);
16    // else
17    return result + get_body_power(y, x);
18 }
19
20 int main() {
21     long long x, y;
22     scanf("%lld %lld", &x, &y);
23     printf("%lld\n", get_body_power(y, x));
24     return 0;
25 }
26

```

./1082.波兰表达式.c

```

1 // 9:47
2 // 10:20
3
4 #include <stdio.h>
5 #include <string.h>
6 #include <stdlib.h>
7 #include <ctype.h>
8
9 char *current_pos;
10 char expression[151];
11
12 double calculate(char *str) {
13     char *operator_;
14     operator_ = strtok(str, " ");
15     current_pos = operator_ + strlen(operator_) + 1;
16     if (strlen(operator_) > 1 || isdigit(*operator_)) { // 说明是个数字
17         return atof(operator_);
18     }
19     double operand1 = calculate(current_pos);
20     double operand2 = calculate(current_pos);
21     switch (*operator_) {
22         case '+':
23             return operand1 + operand2;
24         case '-':
25             return operand1 - operand2;
26         case '*':
27             return operand1 * operand2;
28         case '/':
29             return operand1 / operand2;

```

```

30     }
31 }
32
33 int main(int argc, char *argv[]) {
34     int t;
35     scanf("%d", &t);
36     getchar(); // 去掉换行
37     for (int i = 0; i < t; i++) {
38         fgets(expression, 150, stdin);
39         printf("case #d:\n%.6lf\n", i, calculate(expression));
40     }
41     return 0;
42 }
43

```

./1083.地铁站.c

```

1 // 10:26
2 // 10:53
3
4 #include <stdio.h>
5
6 long long board_people[21] = {0}, unboard_people[21] = {0},
7 people_boards[21] = {0};
8 // t是第二站上下车的人数
9 long long get_last_unboard_people(int a, int t, int n) {
10     board_people[1] = a;
11     people_boards[1] = a;
12     board_people[2] = t;
13     unboard_people[2] = t;
14     people_boards[2] = a;
15     for (int i = 3; i < n; i++) {
16         board_people[i] = board_people[i - 1] + board_people[i - 2];
17         unboard_people[i] = board_people[i - 1];
18         people_boards[i] = people_boards[i - 1] + board_people[i] -
19         unboard_people[i];
20     }
21     board_people[n] = 0;
22     unboard_people[n] = people_boards[n - 1];
23     people_boards[n] = 0;
24     return unboard_people[n];
25 }
26
27 int main() {
28     int a, n, m, x;
29     scanf("%d %d %d %d", &a, &n, &m, &x);
30     for (int t = 0; t < m; t++) { // t从1开始的话最后一个测试用例就不对。
31         // 说好的第2站有人上下车呢？ 还能有0人上下车的？
32         if (m == get_last_unboard_people(a, t, n))
33             break;
34     }
35     printf("%lld\n", people_boards[x]);
36 }

```

./1084.Fj&haozi.c

```

1 // 8:09
2 // 8:50
3
4 #include<stdio.h>
5 #include<string.h>
6 #define MAX_N 101 // 原来写的21导致出错
7
8 long long previous_num[MAX_N] = {0};
9 long long current_num[MAX_N] = {0};
10
11 void expand_num(int n, int m) {
12     for (int i = 0; i < m; i++) {
13         for (int j = 2; j < n; j++) {
14             current_num[j] = previous_num[j - 1] + previous_num[j + 1];
15         }
16         current_num[1] = previous_num[2];
17         current_num[n] = previous_num[n - 1];
18         memcpy(previous_num, current_num, MAX_N * sizeof(long long));
19         memset(current_num, 0, MAX_N * sizeof(long long));
20     }
21 }
22
23 int main() {
24     int n, m, p, t;
25     int cas;
26     scanf("%d", &cas);
27     while(cas--) {
28         scanf("%d %d %d %d", &n, &p, &m, &t);
29         memset(previous_num, 0, MAX_N * sizeof(long long));
30         memset(current_num, 0, MAX_N * sizeof(long long));
31         previous_num[p] = 1;
32         expand_num(n, m);
33         printf("%lld\n", previous_num[t]);
34     }
35 }
36
37

```

./1085.泰波那契数列的前74项.c

```

1 // 8:14
2 // 9:23
3
4 #include<stdio.h>
5
6 long long tribonacci[74] = {0LL, 1LL, 1LL};
7
8 int main(){

```

```

9     for(int i = 3; i < 74; i++){
10         tribonacci[i] = tribonacci[i - 1] + tribonacci[i - 2] + tribonacci[i
- 3];
11     }
12     int t;
13     scanf("%d",&t);
14     for (int i = 0; i < t; i++) {
15         int n;
16         scanf("%d", &n);
17         printf("case #d:\n%lld\n", i, tribonacci[n]);
18     }
19     return 0;
20 }
21

```

./1087.统计字符串个数.c

```

1 // 9:41
2 // 10:01
3
4 #include <stdio.h>
5 #include <string.h>
6 #define substring_width 3
7
8 int end_with_0[substring_width];
9 int end_with_1[substring_width];
10
11 void init() {
12     memset(end_with_0, 0, substring_width * sizeof(end_with_0[0]));
13     memset(end_with_1, 0, substring_width * sizeof(end_with_1[0]));
14 }
15
16 void expand(int n) {
17     end_with_0[0] = 0;
18     end_with_1[0] = 0;
19     end_with_0[1] = 1;
20     end_with_1[1] = 1;
21     for (int i = 2; i < n + 1; i++) {
22         end_with_0[2] = end_with_0[1] + end_with_1[1];
23         end_with_1[2] = end_with_0[2] - end_with_1[0];
24         memmove(end_with_0, end_with_0 + 1, substring_width *
sizeof(end_with_0[0]));
25         memmove(end_with_1, end_with_1 + 1, substring_width *
sizeof(end_with_1[0]));
26     }
27 }
28
29 int main() {
30     int n = 0;
31     while (1) {
32         scanf("%d", &n);
33         if (n < 0)
34             break;
35         init();

```

```

36     expand(n);
37     printf("%d\n", end_with_0[1] + end_with_1[1]);
38 }
39 return 0;
40 }
41

```

./1090.最短路径.c

```

1 // 9:19
2 // 9:39
3
4 #include<stdio.h>
5 #define MAX_N 200
6
7 int matrix[MAX_N][MAX_N];
8 int loss[MAX_N][MAX_N];
9 char route[MAX_N][MAX_N];
10
11 void expand_route(int m, int n) {
12     loss[0][0] = matrix[0][0];
13     for (int i = 1; i < m; i++) {
14         loss[i][0] = loss[i - 1][0] + matrix[i][0];
15         route[i][0] = 'D';
16     }
17     for (int i = 0; i < n; i++) {
18         loss[0][i] = loss[0][i - 1] + matrix[0][i];
19         route[0][i] = 'R';
20     }
21     for (int i = 1; i < m; i++) {
22         for (int j = 1; j < n; j++) {
23             if (loss[i][j - 1] < loss[i - 1][j]) {
24                 loss[i][j] = loss[i][j - 1] + matrix[i][j];
25                 route[i][j] = 'R';
26             }
27             else {
28                 loss[i][j] = loss[i - 1][j] + matrix[i][j];
29                 route[i][j] = 'D';
30             }
31         }
32     }
33 }
34
35 void output_route(int i, int j) {
36     if (i == 0 & j == 0) {
37         return;
38     }
39     if (route[i][j] == 'R') {
40         output_route(i, j - 1);
41         putchar(route[i][j]);
42     } else {
43         output_route(i - 1, j);
44         putchar(route[i][j]);
45     }

```

```

46 }
47
48 int main() {
49     int m, n;
50     scanf("%d%d", &m, &n);
51     for (int i = 0; i < m; i++) {
52         for (int j = 0; j < n; j++) {
53             scanf("%d", &matrix[i][j]);
54         }
55     }
56     expand_route(m, n);
57     printf("%d\n", loss[m - 1][n - 1]);
58     output_route(m - 1, n - 1);
59 }
60

```

./1093.波浪图.c

```

1 // 15:00
2 // 15:23
3
4 #include<stdio.h>
5 #include<string.h>
6 #define N 80
7
8 typedef char Picture[2 * N][2 * N];
9
10 void get_pic(Picture pic, char *s) {
11     memset(pic, ' ', sizeof(Picture));
12     int row = N, col = 0;
13     pic[row][col] = *s;
14     for (s++; *s; s++) {
15         col++;
16         if (*s > *(s - 1)) row--;
17         if (*s < *(s - 1)) row++;
18         pic[row][col] = *s;
19     }
20 }
21
22
23 void output(Picture pic, int len) {
24     char blank[2 * N];
25     memset(blank, ' ', sizeof(blank));
26     int start_row = 0, end_row = 2 * N - 1;
27     for (; memcmp(pic[start_row], blank, sizeof(blank)) == 0; start_row++);
28     for (; memcmp(pic[end_row], blank, sizeof(blank)) == 0; end_row--);
29     for (int i = start_row; i <= end_row; i++) {
30         for (int j = 0; j < len; j++) {
31             putchar(pic[i][j]);
32         }
33         putchar('\n');
34     }
35 }
36

```

```

37 int main() {
38     char s[N + 1];
39     Picture pic;
40     while (scanf("%s", s) != EOF) {
41         get_pic(pic, s);
42         output(pic, strlen(s));
43     }
44 }
45
46

```

./1094.坏掉的彩灯.c

```

1 // 15:07
2 // 16:19
3
4 // 18:38
5 // 18:51
6
7 // 19:18
8 // 19:19
9
10 #include<stdio.h>
11 #include<string.h>
12 #define N 100
13
14 char Colour[] = "RBYG";
15 char Index[128];
16
17 void init() {
18     Index['R'] = 0;
19     Index['B'] = 1;
20     Index['Y'] = 2;
21     Index['G'] = 3;
22 }
23
24 int global_valid = 0;
25
26 int get_possible_solution(char s[], char *result, int len) {
27     // int len = strlen(s);
28     int i = 0;
29     for (; i < len; i++) {
30         if (s[i] == '!') break;
31     }
32     // printf("%s\n%d %d\n", result, i, len);
33     if (i == len) {
34         global_valid = 1;
35         memcpy(result, s, (N + 1) * sizeof(char));
36         return 1;
37     }
38     int possible[] = {1, 1, 1, 1}; // RBYG
39     for (int j = i - 3; j <= i + 3; j++) {
40         if (j >= 0 && j < len && s[j] != '!') {
41             possible[Index[s[j]]] = 0;

```

```

42     }
43 }
44 int valid = 0;
45 for (int j = 0; j < 4; j++)
46     if (possible[j]) valid = 1;
47 if (valid == 0) {
48 //     memcpy(result, s, N); // 还原
49     return 0; // 表明这种情况不可能
50 }
51 for (int j = 0; j < 4; j++) {
52     if (possible[j]) {
53         char temp[N + 1];
54         memcpy(result, s, (N + 1) * sizeof(char));
55         result[i] = Colour[j];
56         memcpy(temp, result, (N + 1) * sizeof(char));
57         global_valid = get_possible_solution(temp, result, len);
58         if (global_valid) return 1;
59     }
60 }
61 return 0;
62 }
63
64 int get_broken_nums(char good[], char broken[], int colors[], int len) {
65 //     int len = strlen(broken);
66     for (int i = 0; i < len; i++) {
67         if (broken[i] == '!') {
68             colors[Index[good[i]]] += 1;
69         }
70     }
71 }
72
73 int main() {
74     init();
75     int t;
76     scanf("%d", &t);
77     for (int i = 0; i < t; i++) {
78         char s[N + 1], result[N + 1];
79         scanf("%s", s);
80         int len = strlen(s);
81         memcpy(result, s, (N + 1) * sizeof(char));
82         global_valid = 0;
83         int ret = get_possible_solution(s, result, len);
84         int colors[4] = {0};
85         get_broken_nums(result, s, colors, len);
86         printf("case #d:\n", i);
87         for (int j = 0; j < 3; j++) {
88             printf("%d ", colors[j]);
89         }
90         printf("%d\n", colors[3]);
91     }
92 }
93
94

```


./A.进制转换.c

```
1 #include<stdio.h>
2
3 char str[] = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";
4
5 int main(){
6     int t, n, r;
7     char result[33];
8     scanf("%d", &t);
9     for (int i=0; i<t; i++){
10        scanf("%d %d", &n, &r);
11        if (n < 0) {
12            n = - n;
13            printf("-");
14        }
15        int j = 0;
16        for (; n; n/=r, j++) result[j] = str[n%r];
17        while (--j >= 0) printf("%c", result[j]);
18        printf("\n");
19    }
20 }
21
```

./B.神秘信息.c

```
1 #include<stdio.h>
2
3 int main(){
4     int a[128];
5     int t, digit, n;
6     long long result;
7     char s[61], *p;
8     scanf("%d", &t);
9     for (int i=0; i<t; i++) {
10        printf("case #%d:\n", i);
11        for (int i=0; i<128; i++) a[i] = -1;
12        scanf("%s", s);
13        a[*s] = -1;
14        n = 1;
15        p = s;
16        digit = 0;
17        a[*p] = 1;
18        while (*++p){
19            if (a[*p] != -1) continue;
20            a[*p] = digit;
21            digit = digit ? digit + 1 : 2;
22            n ++;
23        }
24        if (n < 2) n = 2;
25        p = s;
26        result = 0;
```

```

27     while (*p)
28         result = result * n + a[*p++];
29     printf("%lld\n", result);
30 }
31 }
32

```

./C.按数据中1的位数排序.c

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  # define BIT_NUM 64
5  # define MAX_N 10000
6  typedef struct
7  {
8      long long value;
9      int number;
10 } Data;
11
12 int cmp(const void *pa, const void *pb){
13     Data a = *(Data*)pa;
14     Data b = *(Data*)pb;
15     if (a.number == b.number) {
16         if (a.value == b.value)
17             return 0;
18         else
19             return a.value > b.value ? 1 : -1;
20     } else
21         return b.number - a.number;
22 }
23
24 int main() {
25     int t, n;
26     Data nums[MAX_N];
27     scanf("%d", &t);
28     for (int i=0; i<t; i++) {
29         printf("case #d:\n", i);
30         scanf("%d", &n);
31         for (int j=0; j<n; j++) {
32             scanf("%lld", &nums[j].value);
33             nums[j].number = 0;
34             long long d = 1;
35             for (int k=0; k<BIT_NUM; k++) {
36                 if (nums[j].value & d) nums[j].number++;
37                 d <<= 1;
38             }
39         }
40         qsort(nums, n, sizeof(nums[0]), cmp);
41         for (int j=0; j<n; j++)
42             printf("%lld ", nums[j].value);
43         printf("\n");
44     }
45 }

```

./D.内存显示.c

```
1 #include<stdio.h>
2 #include<string.h>
3 #include<stdlib.h>
4
5 void solveint(int n){
6     for (int i=0; i<sizeof(n); i++){
7         printf("%02x ", ((unsigned char*)&n)[i]);
8     }
9 }
10
11 void solvedouble(double d){
12     for (int i=0; i<sizeof(d); i++){
13         printf("%02x ", ((unsigned char*)&d)[i]);
14     }
15 }
16
17 int main(){
18     char s[31];
19     while (scanf("%s", s) != EOF){
20         if (strchr(s, '.') == 0)
21             solveint(atoi(s));
22         else
23             solvedouble(atof(s));
24         printf("\n");
25     }
26     return 0;
27 }
28
```

./E.平衡三进制.c

```
1 #include<stdio.h>
2
3 int main(){
4     int t;
5     char s[30];
6     scanf("%d", &t);
7     for (int i=0; i<t; i++){
8         printf("case #%d:\n", i);
9         scanf("%s", s);
10        int result = 0;
11        for (int j=0; s[j]; j++){
12            switch (s[j]){
13                case '-':
14                    result = result * 3 - 1;
15                break;
16                case '0':
17                    result = result * 3;
```

```

18         break;
19         case '1':
20             result = result * 3 + 1;
21             break;
22     }
23 }
24 printf("%d\n", result);
25 }
26 }
27

```

./F.非重复二进制串.c

```

1  #include<stdio.h>
2
3  int main(){
4      int t;
5      scanf("%d", &t);
6      for (int i=0; i<t; i++){
7          int n;
8          printf("case #%d:\n", i);
9          scanf("%d", &n);
10         int max_diff_bin_len = 1, diff_bin_len = 1; // 初始值应为1, 因为如果大于
// 0的话, 一位肯定是不重复的。
11         for (; n > 1; n >>= 1){ // 最左的01不能算, 因为前置0不计算在内, 所以右移到01
// 时就可以停止了, 大于1代表可以继续右移。
12             switch (n & 0b11){
13                 case 0b10:
14                 case 0b01:
15                     diff_bin_len ++;
16                     if (diff_bin_len > max_diff_bin_len)
17                         max_diff_bin_len = diff_bin_len;
18                     break;
19                 case 0b00:
20                 case 0b11:
21                     diff_bin_len = 1;
22             }
23         }
24         printf("%d\n", max_diff_bin_len);
25     }
26 }
27

```

./G.二进制位不同的个数.c

```

1  #include<stdio.h>
2
3  int main(){
4      int t;
5      scanf("%d", &t);
6      for (int i=0; i<t; i++){
7          int x, y;

```

```

8     scanf("%d %d", &x, &y);
9     int distance = 0;
10    int xor = x ^ y;
11    for (; xor > 0; xor >>= 1){
12        distance += xor & 1;
13    }
14    printf("%d\n", distance);
15 }
16 }
17

```

./H.数据密度.c

```

1  #include<stdio.h>
2  #define MAX_LEN 130
3
4  int bin_with_1(char *pc){
5      unsigned char c = *(unsigned char*)pc; // 中文的每个字节首位可能是1
6      int result = 0;
7      do{
8          result += c & 1;
9      }while (c >>= 1);
10     return result;
11 }
12
13 /* 只能计算b > a情况时的最大公因数 */
14 int gcd(int a, int b){
15     if (a <= 0) return b;
16     int temp = b % a;
17     return gcd(temp, a);
18 }
19
20 int main(){
21     char s[500], *pchar;
22     int n;
23     scanf("%d", &n);
24     getchar(); // 因为scanf不会读取换行符, 所以要手动把第一个换行符读取了
25     while (n--){
26         fgets(s, MAX_LEN, stdin); // 会把结尾的换行符也读进来
27         // getline(&s, MAX_LEN, stdin);
28         int sum_with_bin_1 = 0;
29         for (pchar = s; (*pchar != '\n') && *pchar; pchar++){
30             sum_with_bin_1 += bin_with_1(pchar);
31         }
32         int sum_of_bin = (pchar - s) * 8;
33         int divisor = gcd(sum_with_bin_1, sum_of_bin);
34         printf("%d/%d\n", sum_with_bin_1 / divisor, sum_of_bin / divisor);
35     }
36 }
37

```

./I.QR Code.c

```

1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<string.h>
4  #include<malloc.h>
5  #define NUM_SIGN "0001"
6  #define MAX_LEN 500
7  #define GROUP_LEN 3
8
9
10 void itob_print(int a, int bitlen){
11     char *result = (char*)malloc(bitlen + 1);
12     result[bitlen] = '\0';
13     while (--bitlen >= 0){
14         result[bitlen] = (a & 1) + '0';
15         a >>= 1;
16     }
17     printf(result);
18     free(result);
19 }
20
21 int main(){
22     char s[MAX_LEN + 1], *p_char;
23     scanf("%s", s);
24     printf(NUM_SIGN);
25     itob_print(strlen(s), 10);
26     int part_len=0, part_num=0;
27     for (p_char = s; *p_char; p_char++){
28         part_len ++;
29         part_num = part_num * 10 + *p_char - '0';
30         if (part_len >= GROUP_LEN){ // 每组的数字已完成
31             itob_print(part_num, 10);
32             part_len = 0;
33             part_num = 0;
34         }
35     }
36     switch (part_len){ // 还有剩余的一些数字没输出
37         case 1:
38             itob_print(part_num, 4);
39             break;
40         case 2:
41             itob_print(part_num, 7);
42             break;
43     }
44 }
45 }
46

```

./J.i-1进制 (Easy) .c

```

1  // 实部
2  // sqrt s
3  // 0 s2 2 2s2 4 4s2 8 8s2 16 16s2
4  // 虚部

```

```

5 // 0 3 6 9 12 15 18 21 24 27 30
6 // 0 3 6 1 4 7 2 5 0 3 6
7 // 18:04
8 #include<stdio.h>
9 #include<string.h>
10
11 typedef struct {
12     long long real;
13     long long imaginary;
14 } Complex;
15
16 /* 计算-1 + i的幂 */
17 Complex pow_of_complex(int multiple_times) {
18     // if (multiple_times <= 0) return {1LL, 0LL};
19     long long modulus = 1LL << (multiple_times / 2); // 偶数时是模长的意思, 奇数
    时是实部或虚部的绝对值 (实部和虚部绝对值相等)
20     Complex result = {0LL, 0LL};
21     switch (multiple_times % 8) {
22         case 0: // 0 * 45°
23             result.real = modulus;
24             break;
25         case 1: // 3 * 45°
26             result.real = - modulus;
27             result.imaginary = modulus;
28             break;
29         case 2: // 6 * 45°
30             result.imaginary = - modulus;
31             break;
32         case 3: // 1 * 45°
33             result.real = modulus;
34             result.imaginary = modulus;
35             break;
36         case 4: // 4 * 45°
37             result.real = - modulus;
38             break;
39         case 5: // 7 * 45°
40             result.real = modulus;
41             result.imaginary = - modulus;
42             break;
43         case 6: // 2 * 45°
44             result.imaginary = modulus;
45             break;
46         case 7: // 5 * 45°
47             result.real = - modulus;
48             result.imaginary = - modulus;
49             break;
50     }
51     return result;
52 }
53
54 /* 两个复数即使在不需要二进制进位的情况下相加, 也不能直接使用异或, 因为负数的时候异或就不对了 */
55 Complex add(Complex a, Complex b) {
56     Complex result;
57     result.real = a.real + b.real;

```

```

58     result.imaginary = a.imaginary + b.imaginary;
59     return result;
60 }
61
62 int main(){
63     char str_hex_num[100];
64     Complex result = {0LL, 0LL};
65     scanf("0x%s", str_hex_num);
66     char *pos_hex_num = str_hex_num;
67     for (int weight=strlen(str_hex_num) - 1; weight>=0; weight--,
pos_hex_num++){
68         int hex_num;
69         if (*pos_hex_num <= '9')
70             hex_num = *pos_hex_num - '0';
71         else
72             hex_num = *pos_hex_num - 'A' + 10;
73         for (int i=0; hex_num>0; hex_num >>= 1, i++){
74             if (hex_num & 1)
75                 result = add(result, pow_of_complex(4 * weight + i));
76         }
77     }
78     if (result.real > 0 && 4 * strlen(str_hex_num) >= 128 && (*str_hex_num
== '1' || *str_hex_num == 'E'))
79         result.real --; // 看了测试用例后发现超出范围的都是输出比答案多了1, 所以手动判
断是否超出范围
80     if (result.real != 0 && result.imaginary > 0 && 4 * strlen(str_hex_num)
>= 128 && (*str_hex_num == '3' || *str_hex_num == '7' || *str_hex_num ==
'E'))
81         result.imaginary --; // 看了测试用例后发现超出范围的都是输出比答案多了1, 所以
手动判断是否超出范围
82     if (result.imaginary < 0 && 4 * strlen(str_hex_num) >= 128 &&
(*str_hex_num == '7' ))
83         result.imaginary ++; // 看了测试用例后发现超出范围的都是输出比答案多了1, 所以
手动判断是否超出范围
84     if (result.imaginary == 0) {
85         printf("%lld", result.real);
86     }else if (result.real == 0 ) {
87         if (result.imaginary == 1) printf("i");
88         else if (result.imaginary == -1) printf("-i");
89         else printf("%lldi", result.imaginary);
90     }else{
91         if (result.imaginary == 1) printf("%lld+i", result.real);
92         else if (result.imaginary == -1) printf("%lld-i", result.real);
93         else printf("%lld%lldi", result.real, result.imaginary);
94     }
95     return 0;
96 }
97 // 20:31
98

```

./K.-1+i进制.c

```

1 // 21:13
2 #include<stdio.h>

```



```

3 #include<regex.h>
4 #include<string.h>
5 #include<stdlib.h>
6 #include<malloc.h>
7 #define MAX_E 50 // 最大位数的整数
8 #define MAX_LINE 100 // 输入的一行最大字符
9
10 typedef struct {
11     long long real;
12     long long imaginary;
13 } Complex;
14
15 regex_t reg_full, reg_real, reg_imaginary;
16 regmatch_t matches[10];
17 int ret = 0;
18 int cflags = REG_EXTENDED | REG_ICASE | REG_NEWLINE;
19 char input[MAX_LINE] = {'\0'};
20
21 int getsign(char *input, regmatch_t *match) {
22     // printf("%s %d %d\n", input, match->rm_so, match->rm_eo);
23     if (match->rm_so < 0 || match->rm_so == match->rm_eo) // 省略正号
24         return 1;
25     else if (*(input + match->rm_so) == '+')
26         return 1;
27     return -1;
28 }
29
30 long long getabs(char *input, regmatch_t *match) {
31     // printf("%s %d %d\n", input, match->rm_so, match->rm_eo);
32     if (match->rm_so < 0 || match->rm_so == match->rm_eo) // 省略1
33         return 1;
34     char *abs_str = malloc((match->rm_eo - match->rm_so + 1) *
sizeof(char));
35     int i;
36     for (i = match->rm_so; i < match->rm_eo; i++) {
37         abs_str[i - match->rm_so] = input[i];
38     }
39     abs_str[i - match->rm_so] = '\0'; // 之前这里少了 - match->rm_so
40     // char *abs_str = strdup(input + match->rm_so, match->rm_eo - match-
>rm_so); // 使用这个在本地运行不会报错, 在OJ运行会Runtime error: SIGSEGV
41     long long result = atoll(abs_str);
42     free(abs_str);
43     return result;
44 }
45
46 Complex get_complex(char *input, int limit) {
47     Complex input_complex = {0LL, 0LL};
48     fgets(input, limit, stdin);
49     ret = regexec(&reg_full, input, 5, matches, 0);
50     if (ret) { // 省略了实部或虚部
51         if (strchr(input, 'i') == NULL) { // 只有实部
52             ret = regexec(&reg_real, input, 3, matches, 0);
53             input_complex.real = getsign(input, matches + 1)
* getabs(input, matches + 2);
54         } else { // 只有虚部

```

```

56         ret = regexec(&reg_imaginary, input, 3, matches, 0);
57         input_complex.imaginary = getsign(input, matches + 1)
58             * getabs(input, matches + 2);
59     }
60 } else {
61     input_complex.real = getsign(input, matches + 1)
62         * getabs(input, matches + 2);
63     input_complex.imaginary = getsign(input, matches + 3)
64         * getabs(input, matches + 4);
65 }
66 return input_complex;
67 }
68
69 Complex divide_once(Complex a) {
70     Complex result;
71     // >>1是向下取整, /2是向0取整
72     result.real = a.imaginary / 2 - a.real / 2;
73     result.imaginary = - a.real / 2 - a.imaginary / 2;
74     if (a.real & 1) { // 这里已经确保实部虚部奇偶性相同, 实部虚部都为奇数时上面两行会
75         // 偏差1, 对负数用%2会出现问题, 所以应用&1来判断奇偶
76         if (a.real < 0 && a.imaginary < 0)
77             result.imaginary ++; // 因为向0取整, 所以向远离0的地方偏移1
78         if (a.real > 0 && a.imaginary > 0)
79             result.imaginary --;
80         if (a.real < 0 && a.imaginary > 0)
81             result.real ++; // 异号时实部会有偏差
82         if (a.real > 0 && a.imaginary < 0)
83             result.real --;
84     }
85     return result;
86 }
87
88 void print_bin_complex(Complex a) {
89     if (a.imaginary == 0 && (a.real == 1 || a.real == 0)){
90         printf("%lld", a.real);
91         return;
92     } else if ((a.real & 1) != (a.imaginary & 1)) { // 负奇数对2取余是-1, 因此
93         // 要取绝对值, 但用&1来判断就不用取绝对值了
94         a.real --;
95         print_bin_complex(divide_once(a));
96         printf("1");
97     } else {
98         print_bin_complex(divide_once(a));
99         printf("0");
100     }
101 }
102
103 // 21:42
104 // 11:00
105 // 11:51
106 // 13:49
107 // 17:03
108 // 19:20
109 // 20:38
110
111 int main(){

```

```

109 // Complex input_complex = {"0", POS}, {"0", POS}};
110 regcomp(&reg_full, "(-)?([0-9]+)(\\+|-)([0-9]*)i$", cflags); // 加入
REG_NEWLINE使^$忽略换行符, 不加^来忽略可能不省略的分号
111 regcomp(&reg_real, "(-)?([0-9]+)$", cflags); // 加入REG_NEWLINE使^$忽略换
行符
112 regcomp(&reg_imaginary, "(-)?([0-9]*)i$", cflags); // 加入REG_NEWLINE使
^$忽略换行符
113 Complex input_complex = get_complex(input, MAX_LINE - 1);
114 // printf("%lld %lld\n", input_complex.real, input_complex.imaginary);
115 // Complex temp_complex = get_complex(input, MAX_LINE);
116 print_bin_complex(input_complex);
117 }
118

```

./K.-1+i进制 (长整数加减法未完成) .c

```

1 // 21:13
2 #include<stdio.h>
3 #include <sys/types.h>
4 #include <regex.h>
5 #include<string.h>
6 #define MAX_E 50 // 最大位数的整数
7 #define MAX_LINE 100 // 输入的一行最大字符
8
9 typedef struct {
10     char *abs; // char abs[MAX_E]
11     enum {NEG=-1, POS=1} sign;
12 } Decimal;
13
14 typedef struct {
15     Decimal real;
16     Decimal imaginary;
17 } Complex;
18
19 regex_t reg_full, reg_real, reg_imaginary;
20
21 Decimal add(Decimal a, Decimal b) {
22     char abs[MAX_E] = "0";
23     Decimal result={abs, POS};
24     if (a.sign == b.sign) {
25         result.sign = a.sign;
26         for (int i=MAX_E; i>0; i--) {
27             int temp = a.abs[i] + b.abs[i];
28             result.abs[i] = temp % 10;
29             result.abs[i - 1] = temp / 10;
30         }
31     } else if (a.abs > b.abs) {
32         result.sign = a.sign;
33         for (int i=MAX_E; i>0; i--) {
34             int temp = a.abs[i] - b.abs[i];
35             result.abs[i] += temp % 10;
36             result.abs[i - 1] += temp / 10;
37         }
38     } else if (a.abs < b.abs) {

```

```

39     result.sign = b.sign;
40     for (int i=MAX_E; i>0; i--) {
41         int temp = b.abs[i] - a.abs[i];
42         result.abs[i] += temp % 10;
43         result.abs[i - 1] += temp / 10;
44     }
45 }
46 return result;
47 }
48
49 int getsign(char *input, regmatch_t *match) {
50     if (match->rm_eo == match->rm_so)
51         return POS;
52     else if (*(input + match->rm_so) == '+')
53         return POS;
54     return NEG;
55 }
56
57 char *getabs(char *input, regmatch_t *match) {
58     if (match->rm_eo == match->rm_so) // 省略1
59         return "1";
60     return strdup(input + match->rm_so, match->rm_eo - match->rm_so);
61 }
62
63 Complex get_complex(char *input, int limit) {
64     int ret;
65     regmatch_t matches[5];
66     Complex input_complex = {{"0", POS}, {"0", POS}};
67     fgets(input, MAX_LINE, stdin);
68     ret = regexec(&reg_full, input, 5, matches, 0);
69     if (ret) { // 省略了实部或虚部
70         if (strchr(input, 'i') == NULL) { // 只有实部
71             ret = regexec(&reg_real, input, 3, matches, 0);
72             input_complex.real.sign = getsign(input, matches + 1);
73             input_complex.real.abs = getabs(input, matches + 2);
74         } else { // 只有虚部
75             ret = regexec(&reg_imaginary, input, 3, matches, 0);
76             input_complex.imaginary.sign = getsign(input, matches + 1);
77             input_complex.imaginary.abs = getabs(input, matches + 2);
78         }
79     } else {
80         input_complex.real.sign = getsign(input, matches + 1);
81         input_complex.real.abs = getabs(input, matches + 2);
82         input_complex.imaginary.sign = getsign(input, matches + 3);
83         input_complex.imaginary.abs = getabs(input, matches + 4);
84     }
85     return input_complex;
86 }
87
88 // 21:42
89 // 11:00
90 // 11:51
91 // 13:49
92 int main(){
93     int ret;

```

```

94     int cflags = REG_EXTENDED | REG_ICASE | REG_NEWLINE;
95     char input[MAX_LINE];
96     // Complex input_complex = {"0", POS}, {"0", POS}};
97     regcomp(&reg_full, "(-)?([0-9]+)(\\+|-)([0-9]*)i$", cflags); // 加入
REG_NEWLINE使^$忽略换行符, 不加^来忽略可能不省略的分号
98     regcomp(&reg_real, "(-)?([0-9]+)$", cflags); // 加入REG_NEWLINE使^$忽略换
行符
99     regcomp(&reg_imaginary, "(-)?([0-9]*)i$", cflags); // 加入REG_NEWLINE使
^$忽略换行符
100    Complex input_complex = get_complex(input, MAX_LINE);
101    printf("%d %s %d %s\n",
102           input_complex.real.sign, input_complex.real.abs,
103           input_complex.imaginary.sign, input_complex.imaginary.abs);
104    // Complex temp_complex = get_complex(input, MAX_LINE);
105    printf("%d %s\n",
106           add(input_complex.real, input_complex.imaginary).sign,
107           add(input_complex.imaginary, input_complex.real).abs);
108 }
109

```

./L.平衡三进制.c

```

1 // 9:37
2 // >>> import math
3 // >>> math.log(3**30, 10)
4 // 14.313637641589873
5
6 #include<stdio.h>
7 #include<regex.h>
8 #include<stdlib.h>
9 #define MAX_LEN 50
10 #define BASE 3
11
12 /* 计算a < b时a和b的最大公因数 */
13 long long gcd(long long a, long long b) {
14     if (a == 0) return b;
15     return gcd(b % a, a);
16 }
17
18 long long tri_to_ll(char *a, int len) {
19     long long result = 0;
20     for (char *pos_num = a; pos_num < a + len; pos_num++) {
21         int num = *pos_num - '0';
22         if (num == 2) num = -1;
23         result = result * BASE + num;
24     }
25     return result;
26 }
27
28 int main() {
29     char tri[MAX_LEN];
30     fgets(tri, MAX_LEN - 1, stdin);
31     regex_t reg;
32     regmatch_t matches[3];

```

```

33     int cflags = REG_NEWLINE | REG_EXTENDED | REG_ICASE;
34     regcomp(&reg, "[0-9+}\\\\.?([0-9]*)", cflags);
35     int ret = regexec(&reg, tri, 3, matches, 0);
36     // for (int i=1; i<3; i++) {
37     //     if (matches[i].rm_so < 0) continue;
38     //     for (int j=matches[i].rm_so; j<matches[i].rm_eo; j++){
39     //         printf("%c", tri[j]);
40     //     }
41     //     printf("\n");
42     // }
43     long long int_part = tri_to_ll(tri + matches[1].rm_so, matches[1].rm_eo
- matches[1].rm_so);
44     long long numerator = tri_to_ll(tri + matches[2].rm_so, matches[2].rm_eo
- matches[2].rm_so);
45     long long denominator = 1;
46     for (int point_pos = matches[2].rm_eo - matches[2].rm_so; point_pos > 0;
point_pos--) {
47         denominator *= 3;
48     }
49     if (int_part > 0 && numerator < 0) { // 借位
50         int_part --;
51         numerator += denominator;
52     }
53     if (int_part < 0 && numerator > 0) { // 借位
54         int_part ++;
55         numerator = denominator - numerator;
56     }
57     long long factor = gcd(llabs(numerator), llabs(denominator));
58     if (int_part || numerator==0)
59         printf("%lld ", int_part);
60     if (numerator) {
61         printf("%lld %lld", numerator / factor, denominator / factor);
62     }
63 }
64 // 10:52
65

```

./M.平衡三进制II.c

```

1 // 10:53
2 // >>> math.log(10**9,3)
3 // 18.86312946860446
4 #include<stdio.h>
5 #include<stdlib.h>
6 #include<string.h>
7 #define MAX_LEN 50
8
9 int get_point_pos(long long denominator) {
10     int point_pos = 0;
11     while (denominator > 1) {
12         denominator /= 3;
13         point_pos ++;
14     }
15     return point_pos;

```

```

16 }
17
18 /* 十进制转三进制，返回三进制的字符个数 */
19 int dec_to_tri(long long dec, char *tri) {
20     if (dec < 3) {
21         *tri = dec + '0';
22         return 1;
23     }
24     int len = dec_to_tri(dec / 3, tri);
25     *(tri + len) = dec % 3 + '0';
26     return len + 1;
27 }
28
29 /* 注意，这里的target-1必须是一个合法的地址，用来存放进位 */
30 int add_tri(char *target, char *addtion, int len) {
31     int carry = 0; // 存储进位
32     for (int i = len - 1; i >= 0; i--) {
33         target[i] += addtion[i] - '0' + carry;
34         carry = 0;
35         if (target[i] >= '3') {
36             carry = 1;
37             target[i] -= 3;
38         }
39     }
40     if (carry) // 最后一次进位
41         target[-1] ++;
42 }
43
44 int sub_ones(char *target, int len) {
45     for (char *i = target; i < target + len; i++) {
46         (*i) --;
47         if (*i < '0') *i = '2';
48     }
49 }
50
51 void my_strncpy(char *dest, char *src, int limit) {
52     char *p_dest = dest, *p_src = src;
53     while (*p_src && limit > 0) {
54         *p_dest++ = *p_src++;
55         limit --;
56     }
57     *p_dest = 0;
58 }
59
60 char *simplify_int_part(char *part, int len) {
61     while (len > 1 && *part == '0')
62         part++, len--;
63     return part;
64 }
65
66 int simplify_fraction_part(char *part, int len) {
67     while (len > 0 && part[--len] == '0')
68         part[len] = 0;
69     if (part[len] == '0' || part[len] == 0) return len;
70     else return len + 1;

```

```

71 }
72
73 // 11:43
74 // 12:48
75 // 14:43
76
77 int main() {
78     long long numerator, denominator;
79     scanf("%lld %lld", &numerator, &denominator);
80     char tri[MAX_LEN], ones[MAX_LEN];
81     tri[0] = '0';
82     int len = dec_to_tri(numerator, tri + 1); // 留出第一位可能的进位空间
83     tri[1 + len] = '\0';
84     // printf("%s\n", tri);
85     for (int i = 0; i < len; i++)
86         ones[i] = '1'; // 构造全1序列
87     ones[len] = '\0';
88     add_tri(tri + 1, ones, len);
89     // printf("%s\n", tri);
90     sub_ones(tri + 1, len);
91     // printf("%s\n", tri);
92     int point_pos = get_point_pos(denominator);
93     // char *p_endpos = tri + 1 + len;
94     // char *output = tri;
95     // while (*output == '0') output++; // 去除前面的0
96     // if (output >= p_endpos - point_pos) putchar('0'); // 纯小数
97     // while (output < p_endpos - point_pos) putchar(*output++); // 输出整数
    部分
98     // while (*(p_endpos - 1) == '0') p_endpos--; // 去除后面的0
99     // if (point_pos > 0 && output < p_endpos) { // 判断是否需要输出小数部分
100     //     putchar('.');
101     //     for (int i = point_pos - (p_endpos - output); i > 1; i--) {
102     //         putchar('0'); // 纯小数
103     //     }
104     //     while (output < p_endpos) putchar(*output++);
105     // }
106     int full_len = len + 1; // 三进制数字总长度
107     char tri_with_start_0[MAX_LEN];
108     int i = 0;
109     for (; i < point_pos - full_len + 1; i++) {
110         tri_with_start_0[i] = '0';
111     }
112     int j = 0;
113     while (tri[j]) tri_with_start_0[i++] = tri[j++];
114     tri_with_start_0[i] = 0;
115     full_len = strlen(tri_with_start_0);
116     char int_part[MAX_LEN], fraction_part[MAX_LEN];
117     my_strncpy(int_part, tri_with_start_0, full_len - point_pos);
118     my_strncpy(fraction_part, tri_with_start_0 + full_len - point_pos,
point_pos);
119     char *new_int_part = simplify_int_part(int_part, strlen(int_part));
120     int fraction_len = simplify_fraction_part(fraction_part,
strlen(fraction_part));
121     printf("%s", new_int_part);
122     if (fraction_len > 0)

```



```

123     printf("%.5s", fraction_part);
124     return 0;
125 }
126
127

```

./library/大整数计算1.c

```

1 // https://blog.csdn.net/u013113678/article/details/113060506
2
3 #include<stdint.h>
4 #include<stdio.h>
5 #include<string.h>
6 //计算位数, 范围[1,9]
7 #define NUM_DIGIT 9
8 //数组类型
9 #if NUM_DIGIT<=0
10     static_assert(1, "NUM_DIGIT must > 0 and <=9");
11 #elif NUM_DIGIT <=2
12 #define _NUM_T int8_t
13 #elif NUM_DIGIT <=4
14 #define _NUM_T int16_t
15 #elif NUM_DIGIT <=9
16 //NUM_DIGIT在[5,9]时用int32就可以装载数据, 但是实测发现用int32在32位程序中, 对除法性能有影响, 用int64反而正常, 这是比较奇怪的现象。
17 #define _NUM_T int64_t
18 #else
19     static_assert(1, "NUM_DIGIT must > 0 and <=9");
20 #endif
21 //计算进制
22 static const size_t _hex = NUM_DIGIT == 1 ? 10 : NUM_DIGIT == 2 ? 100 :
NUM_DIGIT == 3 ? 1000 : NUM_DIGIT == 4 ? 10000 : NUM_DIGIT == 5 ? 100000 :
NUM_DIGIT == 6 ? 1000000 : NUM_DIGIT == 7 ? 10000000 : NUM_DIGIT == 8 ?
100000000 : NUM_DIGIT == 9 ? 1000000000 : -1;
23 #define _MIN_NUM_SIZE 30/NUM_DIGIT
24
25     /// <summary>
26     /// 通过字符串初始化
27     /// </summary>
28     /// <param name="a">[in]高精度数组</param>
29     /// <param name="value">[in]字符串首地址</param>
30     /// <param name="len">[in]字符串长度</param>
31     /// <returns>数据长度</returns>
32     static size_t initByStr(_NUM_T* a, const char* value, size_t len)
33     {
34         size_t shift = 10;
35         size_t i, j, k = 0, n;
36         n = len / NUM_DIGIT;
37         for (i = 0; i < n; i++)
38         {
39             a[i] = (value[len - k - 1] - '0');
40             for (j = 1; j < NUM_DIGIT; j++)
41             {
42                 a[i] += (value[len - k - j - 1] - '0') * shift;

```

```

43         shift *= 10;
44     }
45     shift = 10;
46     k += NUM_DIGIT;
47 }
48 if (n = len - n * NUM_DIGIT)
49 {
50     a[i] = (value[len - k - 1] - '0');
51     for (j = 1; j < n; j++)
52     {
53         a[i] += (value[len - k - j - 1] - '0') * shift;
54         shift *= 10;
55     }
56     i++;
57 }
58 return i;
59 }
60 /// <summary>
61 /// 通过无符号32整型初始化
62 /// </summary>
63 /// <param name="a">[in]高精度数组</param>
64 /// <param name="value">[in]整型值</param>
65 /// <returns>数据长度</returns>
66 static size_t initByInt32(_NUM_T* a, uint32_t value)
67 {
68     size_t i;
69     for (i = 0; i < 8096; i++)
70     {
71         a[i] = value % _hex;
72         value /= _hex;
73         if (!value)
74         {
75             i++;
76             break;
77         }
78     }
79     return i;
80 }
81 /// <summary>
82 /// 通过无符号64整型初始化
83 /// </summary>
84 /// <param name="a">[in]高精度数组</param>
85 /// <param name="value">[in]整型值</param>
86 /// <returns>数据长度</returns>
87 static size_t initByInt64(_NUM_T* a, uint64_t value)
88 {
89     size_t i;
90     for (i = 0; i < 8096; i++)
91     {
92         a[i] = value % _hex;
93         value /= _hex;
94         if (!value)
95         {
96             i++;
97             break;

```

```

98     }
99     }
100    return i;
101    }
102    ///

```

```

152     /// <param name="aLen">[in]被加数长度</param>
153     /// <param name="b">[in]加数</param>
154     /// <param name="bLen">[in]加数长度</param>
155     /// <returns>结果长度</returns>
156     static size_t acc(_NUM_T* a, size_t aLen, const _NUM_T* b, size_t
bLen)
157     {
158         size_t i, n = bLen;
159         const size_t max = _hex - 1;
160         if (aLen < bLen)
161         {
162             memset(a + aLen, 0, (bLen - aLen + 1) * sizeof(_NUM_T));
163         }
164         else {
165             a[aLen] = 0;
166         }
167         for (i = 0; i < bLen; i++) {
168             uint64_t temp = (uint64_t)a[i] + (uint64_t)b[i];
169             a[i] = temp % _hex;
170             a[i + 1] += temp / _hex;
171         }
172         for (; i < aLen; i++) {
173             uint64_t temp = a[i];
174             if (temp <= max)
175                 break;
176             a[i] = temp % _hex;
177             a[i + 1] += temp / _hex;
178         }
179         n = aLen > bLen ? aLen : bLen;
180         if (a[n])
181             return n + 1;
182         return n;
183     }
184
185     /// <summary>
186     /// 减法(累减)
187     /// 结果会保存在a中
188     /// </summary>
189     /// <param name="a">[in]被减数, 被减数必须大于等于减数</param>
190     /// <param name="aLen">[in]被减数长度</param>
191     /// <param name="b">[in]减数</param>
192     /// <param name="bLen">[in]减数长度</param>
193     /// <returns>结果长度</returns>
194     static size_t subc(_NUM_T* a, size_t aLen, const _NUM_T* b, size_t
bLen) {
195         size_t m, n, i;
196         if (aLen < bLen)
197         {
198             m = bLen;
199             n = aLen;
200         }
201         else {
202             n = bLen;
203             m = aLen;
204         }

```

```

205     for (i = 0; i < n; i++)
206     {
207         int64_t temp = (int64_t)a[i] - (int64_t)b[i];
208         a[i] = temp;
209         if (temp < 0)
210         {
211             a[i + 1] -= 1;
212             a[i] += _hex;
213         }
214     }
215     for (; i < m; i++)
216     {
217         _NUM_T temp = a[i];
218         if (temp < 0)
219         {
220             a[i + 1] -= 1;
221             a[i] += _hex;
222         }
223         else
224         {
225             break;
226         }
227     }
228     for (size_t i = aLen - 1; i != SIZE_MAX; i--)
229         if (a[i])
230             return i + 1;
231     return 1;
232 }
233
234 /// <summary>
235     /// 乘法
236     /// </summary>
237     /// <param name="a">[in]被乘数</param>
238     /// <param name="aLen">[in]被乘数长度</param>
239     /// <param name="b">[in]乘数</param>
240     /// <param name="bLen">[in]乘数长度</param>
241     /// <param name="c">[out]结果, 数组长度必须大于等于aLen+bLen+1, 且全部元素为
242     /// <returns>结果长度</returns>
243     static size_t multi(const _NUM_T* a, size_t aLen, const _NUM_T* b,
244     size_t bLen, _NUM_T c[]) {
245         _NUM_T d;
246         size_t j;
247         c[aLen + bLen] = 0;
248         for (size_t i = 0; i < aLen; i++)
249         {
250             d = 0;
251             for (j = 0; j < bLen; j++)
252             {
253                 uint64_t temp = (uint64_t)a[i] * (uint64_t)b[j] + c[j + i]
254                 + d;
255                 d = temp / _hex;
256                 c[j + i] = temp % _hex;
257             }
258             if (d)

```

```

257     {
258         c[j + i] = d;
259     }
260 }
261 for (size_t k = aLen + bLen; k != SIZE_MAX; k--)
262     if (c[k])
263         return k + 1;
264 return 1;
265 }
266
267 /// <summary>
268     /// 除法
269     /// </summary>
270     /// <param name="a">[in]被除数,被除数必须大于除数。小于商为0、余数为除数,等于商
271     为1、余数为0,不需要在函数内实现,在外部通过compare就可以做到。</param>
272     /// <param name="aLen">[in]被除数长度</param>
273     /// <param name="b">[in]除数</param>
274     /// <param name="bLen">[in]除数长度</param>
275     /// <param name="c">[out]商,数组长度大于等于aLen,且全部元素为0</param>
276     /// <param name="mod">[out]余数,数组长度大于等于aLen</param>
277     /// <param name="modLen">[out]余数长度</param>
278     /// <param name="temp">[in]临时缓冲区,由外部提供以提高性能,数组长度大于等于
279     aLen+bLen+1</param>
280     /// <returns>商长度</returns>
281     static size_t divide(const _NUM_T* a, size_t aLen, const _NUM_T*
282     b, size_t bLen, _NUM_T* c, _NUM_T* mod, size_t* modLen, _NUM_T* temp) {
283         intptr_t n, l;
284         int equal;
285         memcpy(mod, a, (aLen) * sizeof(_NUM_T));
286         n = aLen - bLen;
287         l = aLen;
288         while (n > -1)
289         {
290             equal = compare(mod + n, l - n, b, bLen);
291             if (equal == -1)
292             {
293                 n--;
294                 if (!mod[l - 1])
295                     l--;
296                 continue;
297             }
298             if (equal == 0)
299             {
300                 c[n]++;
301                 n -= bLen;
302                 l -= bLen;
303                 continue;
304             }
305             uint64_t x;
306             if ((l - n) > bLen)
307             {
308                 x = ((mod[l - 1]) * _hex) / ((uint64_t)b[bLen - 1] + 1);
309                 if (x == 0)
310                 {
311                     x = (mod[l - 2]) / ((uint64_t)b[bLen - 1] + 1);

```

```

309         }
310     }
311     else
312     {
313         x = (mod[l - 1]) / ((uint64_t)b[bLen - 1] + 1);
314     }
315     if (x == 0)
316         x = 1;
317     c[n] += x;
318     if (x == 1)
319     {
320         l = n + subc(mod + n, l - n, b, bLen);
321     }
322     else
323     {
324         _NUM_T num[_MIN_NUM_SIZE];
325         size_t len = initByInt32(num, x);
326         memset(temp, 0, (aLen + bLen + 1) * sizeof(_NUM_T));
327         len = multi(b, bLen, num, len, temp);
328         l = n + subc(mod + n, l - n, temp, len);
329     }
330 }
331 intptr_t i = l - 1;
332 for (; i > -1; i--)
333     if (mod[i])
334         break;
335 if (i == -1)
336 {
337     mod[0] = 0;
338     *modLen = 1;
339 }
340 else
341 {
342     *modLen = i + 1;
343 }
344 if (c[aLen - bLen] != 0)
345     return aLen - bLen + 1;
346 else
347     return aLen - bLen;
348 }
349
350 int sum(){
351     int64_t n;
352     size_t len, len2;
353     _NUM_T num[1024];
354     _NUM_T num2[1024];
355     scanf("%ld", &n);
356     len = initByInt32(num, 0);
357     for (int64_t i = 1; i <= n; i++)
358     {
359         len2 = initByInt64(num2, i);
360         len = acc(num, len, num2, len2);
361     }
362     print(num, len);
363     return 0;

```

```

364 }
365
366 int minus()
367 {
368     _NUM_T a1[8096], a2[8096];
369     char s1[8096], s2[8096];
370     scanf("%s%s", s1, s2);
371     size_t len1, len2;
372     len1 = initByStr(a1, s1, strlen(s1));
373     len2 = initByStr(a2, s2, strlen(s2));
374     len1 = subc(a1, len1, a2, len2);
375     print(a1, len1);
376     return 0;
377 }
378
379 int factorial(){
380     int64_t n;
381     size_t len, len2;
382     _NUM_T num[8096];
383     _NUM_T num2[8096];
384     _NUM_T num3[8096];
385     _NUM_T* p1 = num;
386     _NUM_T* p2 = num3;
387     scanf("%ld", &n);
388     len = initByInt32(num, 1);
389     for (int64_t i = 1; i <= n; i++)
390     {
391         len2 = initByInt64(num2, i);
392         memset(p2, 0, 8096 * sizeof(_NUM_T));
393         len = multi(p1, len, num2, len2, p2);
394         _NUM_T* temp = p1;
395         p1 = p2;
396         p2 = temp;
397     }
398     print(p1, len);
399     return 0;
400 }
401
402 int division()
403 {
404     _NUM_T a1[8096], a2[8096], c[8096], mod[8096], temp[8096];
405     char s1[8096], s2[8096];
406     scanf("%s%s", s1, s2);
407     size_t len1, len2, ml;
408     len1 = initByStr(a1, s1, strlen(s1));
409     len2 = initByStr(a2, s2, strlen(s2));
410     memset(c, 0, 8096 * sizeof(_NUM_T));
411     len1 = divide(a1, len1, a2, len2, c, mod, &ml, temp);
412     print(c, len1);
413     printf("\n");
414     print(mod, ml);
415     return 0;
416 }
417
418 int main() {

```



```
419     printf("求 n 累加和(ja)。用高精度方法, 求 s=1+2+3+.....+n 的精确值(n 以一般整数输入)。 \n");
420     sum();
421     printf("\n两个任意十一位数的减法; (小于二十位) \n");
422     minus();
423     printf("\n求 N! 的值(ni)。用高精度方法, 求 N! 的精确值(N 以一般整数输入)。 \n");
424     factorial();
425     printf("\n给定两个非负整数A, B, 请你计算 A / B的商和余数。 \n");
426     division();
427 }
428
```

./library/大整数计算2.c

```
1 // https://www.cnblogs.com/HappyXie/articles/1927557.html
2
3 #include <stdio.h>
4 #include <stdlib.h>
5 #define MAX 200
6
7 typedef struct
8 {
9     int len;
10    int s[MAX+1];
11 } hp;
12
13 void input(hp *a, int x) //读入数字
14 {
15     int i;
16
17     a->len = 0;
18
19     while (x > 0)
20     {
21         a->s[1 + a->len++] = x % 10;
22         x /= 10;
23     }
24
25     for (i = a->len + 1; i <= MAX; i++)
26         a->s[i] = 0;
27 }
28
29 void input1(hp *a, char *str) //读入字符串
30 {
31     int i, len;
32
33     a->len = 0;
34
35     if (str == NULL)
36         return;
37
38     len = strlen(str);
39
40     while (len > 0)
```

```

41     {
42         a->s[1 + a->len++] = *(str + len - 1) - '0';
43         len--;
44     }
45
46     for (i = a->len + 1; i <= MAX; i++)
47         a->s[i] = 0;
48 }
49
50 void print(hp *y) //打印数字
51 {
52     int i;
53     for (i = y->len; i >= 1; i--)
54         printf("%d", y->s[i]);
55     printf("\n");
56 }
57
58 void add(hp *a, hp *b, hp *c) //高精度加法c = a + b
59 {
60     int i, len;
61
62     for (i = 1; i <= MAX; i++) c->s[i] = 0;
63
64     if (a->len > b->len) len = a->len;
65     else len = b->len;
66
67     for (i = 1; i <= len; i++)
68     {
69         c->s[i] += a->s[i] + b->s[i];
70         if (c->s[i] >= 10)
71         {
72             c->s[i] -= 10;
73             c->s[i+1]++;
74         }
75     }
76
77     if (c->s[len+1] > 0) len++;
78     c->len = len;
79 }
80
81 void subtract(hp *a, hp *b, hp *c) //高精度减法c = a - b
82 {
83     int i, len;
84
85     for (i = 1; i <= MAX; i++) c->s[i] = 0;
86
87     if (a->len > b->len) len = a->len;
88     else len = b->len;
89
90     for (i = 1; i <= len; i++)
91     {
92         c->s[i] += a->s[i] - b->s[i];
93         if (c->s[i] < 0)
94         {
95             c->s[i] += 10;

```

```

96         c->s[i+1]--;
97     }
98 }
99
100 while (len > 1 && c->s[len] == 0) len--;
101 c->len = len;
102 }
103
104 int compare (hp *a, hp *b) //高精度比较
105 {
106     int len;
107
108     if (a->len > b->len) len = a->len;
109     else len = b->len;
110
111     while (len > 0 && a->s[len] == b->s[len]) len--;
112
113     if (len == 0) return 0;
114     else return a->s[len] - b->s[len];
115 }
116
117 void multiply (hp *a, int b, hp *c) //高精度 * 单精度
118 {
119     int i, len;
120
121     for (i = 1; i <= MAX; i++) c->s[i] = 0;
122     len = a->len;
123
124     for (i = 1; i <= len; i++)
125     {
126         c->s[i] += a->s[i] * b;
127         c->s[i+1] += c->s[i] / 10;
128         c->s[i] %= 10;
129     }
130
131     len++;
132     while (c->s[len] >= 10)
133     {
134         c->s[len+1] += c->s[len] / 10;
135         c->s[len] %= 10;
136         len++;
137     }
138
139     while (len > 1 && c->s[len] == 0) len--;
140     c->len = len;
141 }
142
143 void multiplyh (hp *a, hp *b, hp *c) //高精度 * 高精度
144 {
145     int i, j, len;
146
147     for (i = 1; i <= MAX; i++) c->s[i] = 0;
148
149     for (i = 1; i <= a->len; i++)
150     {

```

```

151     for (j = 1; j <= b->len; j++)
152     {
153         c->s[i+j-1] += a->s[i] * b->s[j];
154         c->s[i+j] += c->s[i+j-1] / 10;
155         c->s[i+j-1] %= 10;
156     }
157 }
158
159 len = a->len + b->len + 1;
160 while (len > 1 && c->s[len] == 0) len--;
161 c->len = len;
162 }
163
164 void power(hp *a, int b, hp *c) //高精度乘方  $c = a^b$ 
165 {
166     hp e;
167
168     if (b == 0)
169     {
170         c->len = 1;
171         c->s[1] = 1;
172     }
173     else if (b == 1)
174     {
175         memcpy(c, a, sizeof(hp));
176     }
177     else
178     {
179         power(a, b / 2, &e);
180         multiplyh(&e, &e, c);
181
182         if (b % 2 == 1)
183         {
184             memcpy(&e, c, sizeof(hp));
185             multiplyh(&e, a, c);
186         }
187     }
188 }
189
190 void divide(hp *a, int b, hp *c, int *d) //高精度 / 单精度 {d为余数}
191 {
192     int i, len;
193
194     for (i = 1; i <= MAX; i++) c->s[i] = 0;
195     len = a->len;
196     *d = 0;
197
198     for (i = len; i >= 1; i--)
199     {
200         *d = *d * 10 + a->s[i];
201         c->s[i] = *d / b;
202         *d %= b;
203     }
204
205     while (len > 1 && c->s[len] == 0) len--;

```

```

206     c->len = len;
207 }
208
209 void multiply10(hp *a) //高精度 * 10
210 {
211     int i;
212     for (i = a->len; i >= 1; i--)
213         a->s[i+1] = a->s[i];
214
215     a->s[1] = 0;
216     a->len++;
217     while (a->len > 1 && a->s[a->len] == 0) a->len--;
218 }
219
220 void divideh(hp *a, hp *b, hp *c, hp *d) //高精度 / 高精度{d为余数}
221 {
222     hp e;
223     int i, len;
224
225     for (i = 1; i <= MAX; i++)
226     {
227         c->s[i] = 0;
228         d->s[i] = 0;
229     }
230
231     len = a->len;
232     d->len = 1;
233
234     for (i = len; i >= 1; i--)
235     {
236         multiply10(d);
237         d->s[1] = a->s[i];
238
239         while (compare(d, b) >= 0)
240         {
241             subtract(d, b, &e);
242             *d = e;
243             c->s[i]++;
244         }
245     }
246
247     while (len > 1 && c->s[len] == 0) len--;
248     c->len = len;
249 }
250

```

./他人做法/1009.QR Code.c

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <math.h>
5 int main()
6 {

```

```

7   int i,j,c,d,a,b;char s[501],t[33333],u[3];
8   scanf("%s",s);
9   c=13;
10  int len = strlen(s);
11  for(i=0;i<len;c=c+10)
12  {
13      if(i+3>len) break;
14      for(j=0;j<3;j++,i++)
15      {
16          u[j]=s[i];
17      }
18      u[j]='\0';
19      a=atoi(u);
20      for(d=c+10;a!=0;d--)
21      {
22          t[d]=a%2+'0';a/=2;
23      }
24      for(;d>c;d--) t[d]='\0';
25
26  }
27  b=strlen(s);
28  if((b-i)==2)
29  {
30      for(j=0;j<2;j++,i++)
31      {
32          u[j]=s[i];
33      }
34      u[j]='\0';
35      a=atoi(u);
36      for(d=c+7;a!=0;d--)
37      {
38          t[d]=a%2+'0';a/=2;
39      }
40      for(;d>c;d--) t[d]='\0';
41      t[c+8]='\0';
42  }
43  else if((b-i)==1)
44  {
45      for(j=0;j<1;j++,i++)
46      {
47          u[j]=s[i];
48      }
49      u[j]='\0';
50      a=atoi(u);
51      for(d=c+4;a!=0;d--)
52      {
53          t[d]=a%2+'0';a/=2;
54      }
55      for(;d>c;d--) t[d]='\0';
56      t[c+5]='\0';
57  }
58  else
59  {
60      t[c+1]='\0';
61  }

```

```

62     for(d=13;b!=0;d--)
63     {
64         t[d]=b%2+'0';b/=2;
65     }
66     for(;d>3;d--) t[d]='0';
67     t[0]='0';t[1]='0';t[2]='0';t[3]='1';
68     printf("%s",t);
69     return 0;
70 }
71

```

./他人做法/1064.A-B.c

```

1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  #include <math.h>
5  void Input(int a[])
6  {
7      char s[502];
8      scanf("%s", s);
9      int i = 0, j, k;
10     j = strlen(s) - 1;
11     for (k = 501; j >= 0; j--)
12     {
13         a[k] = s[j] - '0';
14         k--;
15     }
16 }
17 int main()
18 {
19     while (1)
20     {
21         int a[502] = {0};
22         int b[502] = {0};
23         Input(a);
24         Input(b);
25         int m = 0, n = 0;
26         int j = 0;
27         while (a[m] == 0)
28         {
29             m++;
30         }
31         while (b[n] == 0)
32         {
33             n++;
34         }
35         if (n > m)
36         {
37             for (int i = 501; i >= m; i--)
38             {
39                 if (a[i] - b[i] >= 0)
40                 {
41                     a[i] = a[i] - b[i];

```

```
42     }
43     else
44     {
45         a[i] = a[i] + 10 - b[i];
46         a[i - 1]--;
47     }
48 }
49 while (a[j] == 0)
50 {
51     j++;
52 }
53 for (j; j <= 501; j++)
54 {
55     printf("%d", a[j]);
56 }
57 printf("\n");
58 }
59
60 if (m > n)
61 {
62     for (int i = 501; i >= n; i--)
63     {
64         if (b[i] - b[i] >= 0)
65         {
66             b[i] = b[i] - a[i];
67         }
68         else
69         {
70             b[i] = b[i] + 10 - a[i];
71             b[i - 1]--;
72         }
73     }
74     while (b[j] == 0)
75     {
76         j++;
77     }
78     printf("-");
79     for (j; j <= 501; j++)
80     {
81         printf("%d", b[j]);
82     }
83     printf("\n");
84 }
85 else
86 {
87     if (a[m] >= b[n])
88     {
89         for (int i = 501; i >= 0; i--)
90         {
91             if (a[i] - b[i] >= 0)
92             {
93                 a[i] = a[i] - b[i];
94             }
95             else
96             {
```



```

97         a[i] = a[i] + 10 - b[i];
98         a[i - 1]--;
99     }
100 }
101 while (a[j] == 0)
102 {
103     j++;
104 }
105 for (j; j <= 501; j++)
106 {
107     printf("%d", a[j]);
108 }
109 printf("\n");
110 }
111 else
112 {
113     for (int i = 501; i >= 0; i--)
114     {
115         if (b[i] - b[i] >= 0)
116         {
117             b[i] = b[i] - a[i];
118         }
119         else
120         {
121             b[i] = b[i] + 10 - a[i];
122             b[i - 1]--;
123         }
124     }
125     while (b[j] == 0)
126     {
127         j++;
128     }
129     printf("-");
130     for (j; j <= 501; j++)
131     {
132         printf("%d", b[j]);
133     }
134     printf("\n");
135 }
136 }
137 }
138
139 return 0;
140 }
141
142

```

./合并文件.py

```

1 # 21:50
2 import glob
3 import os
4
5 os.chdir("/config/workspace/学校文件/编程思维与实践")

```

```

6
7 with open("合并后.md", "w", encoding='utf-8') as f:
8     files = glob.glob("./**", recursive=True)
9     files.sort()
10    for i in files:
11        if not os.path.isfile(i):
12            continue
13        f.write("# ")
14        print(i, file = f)
15        print("`C", file=f)
16        try:
17            text = open(i, encoding='utf-8').read()
18        except:
19            text = open(i, encoding='gbk').read()
20        f.write(text)
21        print('\n', file=f)
22        print("`", file=f)
23        print('\n', file=f)
24

```

./正则表达式测试.c

```

1  #define _GNU_SOURCE
2  #include <string.h>
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include <sys/types.h>
6  #include <regex.h>
7
8  int main (void)
9  {
10
11     int i;
12     char ebuff[256];
13     int ret;
14     int cflags;
15     regex_t reg;
16     regmatch_t rm[5];
17     char *part_str = NULL;
18
19     cflags = REG_EXTENDED | REG_ICASE;
20
21     char *test_str = "Hello World";
22     char *reg_str = "e(.*?)o";
23
24     ret = regcomp(&reg, reg_str, cflags);
25     if (ret)
26     {
27         regerror(ret, &reg, ebuff, 256);
28         fprintf(stderr, "%s\n", ebuff);
29         goto end;
30     }
31
32     ret = regexec(&reg, test_str, 5, rm, 0);

```

```
33     if (ret)
34     {
35         regerror(ret, &reg, ebuff, 256);
36         fprintf(stderr, "%s\n", ebuff);
37         goto end;
38     }
39
40     regerror(ret, &reg, ebuff, 256);
41     fprintf(stderr, "result is:\n%s\n\n", ebuff);
42
43     for (i=0; i<5; i++)
44     {
45         if (rm[i].rm_so > -1)
46         {
47             part_str = strdup(test_str+rm[i].rm_so, rm[i].rm_eo-
rm[i].rm_so);
48             fprintf(stderr, "%s\n", part_str);
49             free(part_str);
50             part_str = NULL;
51         }
52     }
53
54 end:
55     regfree(&reg);
56
57     return 0;
58 }
59
```