E8-Solution

A 怎么想都进不去吧

题目分析

唯一的难点是输入。使用 `%s+sscanf` 或者 `scanf("[%d:%d]\n")` 或者 `scanf("[%d:%d]")` 都可以。

代码

```
#include <stdio.h>
#define max(a, b) ((a > b) ? a : b)
int main() {
    int i, j;
    while (scanf(" [%d:%d]", &i, &j) > 0) {
        int ans = max(i - j, j - i) + 1;
        if (ans > 32)
            printf("Too Big!\n");
        else
            printf("%d\n", ans);
    }
}
```

B 别忘了说再见

题目分析

这道题主要考察多维数组和字符串匹配。先用二维字符串数组保存告别词，然后对于每个名字使用 `strstr()` 函数进行匹配，输出匹配失败的名字。

示例代码

```
#include <stdio.h>
#include <string.h>

int n, m, flag;// flag为匹配失败的标志
char s[1010][110]; //保存告别词
char name[110];
int main(void) {
    scanf("%d%d ", &n, &m);
    for (int i = 0; i < n; i++)
        gets(s[i]); //按行读入告别词
    for (int i = 0; i < m; i++) {
        scanf("%s", name); //读入名字
        for (int j = 0; j < n; j++) { //与字符串依次匹配
            if (strstr(s[j], name))
```

```
                break;
            if (j == n - 1) //匹配失败并输出对应名字
                printf("%s ", name), flag = 1;
        }
    }
    if (!flag) //匹配失败输出
        puts("Enjoy the scenery and we will meet again.");
    return 0;
}
```

author: Uanu

C 多项式求导

题目分析

本题在思路上较为清晰，即利用数组存储多项式，并利用求导公式进行计算。

本题的易错之处主要在于，部分人在解题过程中，直接用数组下标来表示每一项的指数 k，从而导致数组空间不够（与期中考试多项式相加题目中错误类似）。因此，正确的方式应是用两个数组分别连续存储系数和指数，并进行计算。

示例代码

```
#include <stdio.h>

long long base[110000];     //系数
long long power[110000];    //指数

int main()
{
    int n = 0;
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%lld %lld", &base[i], &power[i]);
        base[i] = base[i] * power[i];
        power[i] -= 1;
    }

    for (int i = 0; i < n; i++) {
        if (base[i] != 0) {
            printf("%lld %lld ", base[i], power[i]);
        }
    }

    return 0;
}
```

D: Rex的隐身

题目分析

输入给出一个字符串 `receive` ，要求我们按照 `R` ， `e` ， `x` 的相对顺序，忽略大小写地进行字符串删除操作。

如果通过事先将给定的字符串 `receive` 全部修改为大写或全部修改为小写来进行对比，则需要注意事先拷贝一次得到 `dst` 。删除操作时同时对 `receive` 和 `dst` 中对应位置的字符串进行操作。

由于题目中给出的字符串 `receive` 仅仅含有大小写英文字母，可以将删去的字符替换为某一类特殊字符，比如: `*` ， `$` 等，删除操作结束后历遍每一个字符并判断是否输出即可。

本题可以用循环做，也可以用 `strchr()` 函数进行处理，需要仔细设计判断条件以避免出错。

```
#include<stdio.h>
#include<string.h>

int main()
{
    char dst[500000+5];//用于拷贝
    char receive[500000+5];//读取字符串
    scanf("%s", receive);
    strcpy(dst, receive);
    int i;
    //大小写统一转换
    for(i=0;receive[i]!='\0';i++)
    {
        receive[i] = tolower(receive[i]);
    }
    char* p1=NULL, *p2, *p3;//定义分别指向r,e,x的指针
    for(p1=strchr(receive,'r');p1!=NULL;p1=strchr(p1+1,'r'))//搜r，结束标志为没
搜到
    {
        p2 = strchr(p1,'e');//从当前位置开始搜e
        if(p2)//搜到了
        {
            p3 = strchr(p2,'x');//从当前位置开始搜x
            if(p3)
                //搜到了符合要求的rex，进行删除
            {   receive[p1-receive]='*';
                receive[p2-receive]='*';
                receive[p3-receive]='*';
                dst[p1-receive]='*';
                dst[p2-receive]='*';
                dst[p3-receive]='*';
             //全部置成特殊字符
```

```
            }
            else
                break;
        }
        else
        {
            break;
        }
    }
    for(i=0;dst[i]!='\0';i++)//遍历，遇到特殊字符跳过，结尾处停止
    {
        if(dst[i]!='*')
            putchar(dst[i]);
    }
    return 0;
}
```

E 魔法币-Magic

考点	难度
数学计算、倒序输出	1

题目分析

这个题其实是整场 E8 中非常简单的一题。实际上也很少有同学在这题出现 bug 。

我们可以很容易的观测到，魔法机器 1 只能产生奇数个魔法币，魔法机器 2 只能产生偶数个魔法币，因此，从结果倒退就行。

如果目前的魔法币数目是偶数，用机器 2，否则使用机器 1，然后把 1 或者 2 存在数组里面，最后倒序输出出来即可。

倒序输出可以使用数组存一下，当然也可以使用递归。

示例程序 1

```
#include <stdio.h>

int n, k = 0, op[40]; // op 用于存指令，k用于指令计数，n是当前魔法币数量

int main()
{
    // 文件读写一直是宋友老师推荐的本地调试方式，可以使用命令行或freopen来达到这个目的
```

```
    // freopen("magic.in", "r", stdin);
    // freopen("magic.out", "w", stdout);
    scanf("%d", &n);
    while (n) // while(n) 就是 while(n!=0) 的意思
        if (n & 1) // if(n&1) 就是 if(n%2!=0) 的意思，说明 n 是奇数
            op[++k] = 1, n /= 2; // 注意 ++k 的作用，这样会先使得 k+=1，然后
op[k]=1
        else
            op[++k] = 2, n = n / 2 - 1; // 注意 ++k 的作用，这样会先使得 k+=1，
然后op[k]=2
    // 经过上面的计算，我们一共进行了 k 次操作了，存在了 op[1] 到 op[k] 的位置上
    while (k) // while(k) 就是 while(k!=0) 的意思
        printf("%d", op[k--]); // 注意这里是 k--，表示先输出 op[k] 然后执行 k-=1
    return 0;
}
```

示例程序 2

```
#include <stdio.h>

// 递归的做法，大家可以通过这个程序再次理解递归。
void solve(int n)
{
    if (!n) return; // if(!n) 就是 if(n==0) 的意思
    if (n & 1)
        solve(n / 2), putchar('1');
    else
        solve(n / 2 - 1), putchar('2');
}

int main()
{
    int n;
    scanf("%d", &n);
    solve(n);
    return 0;
}
```

F 分解，分解，还是分解

题目分析

使用短除法，每次将 n 对 k 取模再将 n 除以 k，直至 n 变成 0 为止。将每次得到的模数存起来，这个模数就是题干中所说的 x_i。最后统一进行输出即可。

标准程序

```
#include <stdio.h>

int main(){
    int n;
    int k;
    scanf("%d%d", &n, &k);
    int num[100];//存储模数
    int len = 0;
    int tem = n;
    while(tem != 0){
        num[len++] = tem % k;
        tem /= k;
    }

    printf("%d=", n);
    for(int i = len - 1; i >= 0; i--){//倒序进行输出
        if(num[i] != 0){
            if(i != len - 1)//第一个输出的数字前没有+
                printf("+");
            printf("%d*%d^%d", num[i], k, i);
        }
    }
    return 0;
}
```

G 魔法实验 3.0

难度	考点
2	构造

问题分析

我们按非空水晶为分界线，将 n 个水晶分为多个空水晶区间，不难发现空区间对答案的影响是独立的。

对于每个空区间，将其按水火间隔的方式附魔必定更优，现在的问题就变为区间的第一个水晶是水还是火。既然对于一个区间只有两种处理方式，直接计算取较小者即可。讨论奇偶性取最优也是可以的。

时间复杂度 $O(f+w)$ 或 $O(n)$ 。

参考代码

```
#include <stdio.h>
#define Min(a, b) ((a) < (b) ? (a) : (b))
int a[100010];
int main()
{
    int n, f, w;
    scanf("%d%d%d", &n, &f, &w);
    int temp;
    for (int i = 0; i < f; i++)
    {
        scanf("%d", &temp);
        a[temp] = 1;
    }
    for (int i = 0; i < w; i++)
    {
        scanf("%d", &temp);
        a[temp] = 2;
    }
    int ans = 0, now = 3, past = 0;
    for (int i = 0; i <= n; i++)
    {
        if (a[i])
            now = a[i];
        else
            now = 3 - now;
        ans += now != past;
        past = now;
    }
    printf("%d", n - ans);
}
```

H 哪吒的区间合并（2）

题目分析

本题与E6H-“哪吒的区间合并（1）”的主要区别有两处，第一是数据范围较大，无法利用数组下标记录区间；第二是本题考虑了区间端点的开闭性，需要一些细节上的变化和处理。基本思路是将区间按照左端点递增进行排序，然后遍历数组根据右端点情况进行合并。具体实现思路如下：

利用二维数组记录区间，二维数组的每行共四个元素，分别记录左端点大小、右端点大小、左端点开闭性、右端点开闭性，其中开闭性可以用 1 表示闭， 0 表示开。

全部读入后，先将二维数组的所有行（即数组记录的所有区间）进行排序，排序规则：按区间左端点大小升序排序，当左端点相同时，闭区间在前，开区间在后。

然后遍历二维数组的每一行，用一个临时一维数组 $t[4]$ 记录当前合并的结果，初始化 t 为当前遍历到的区间，然后继续遍历开始合并区间。只有当前区间左端点小于 t 右端点，或当前区间左端点等于 t 右端点且不全为开区间时，才能合并。

在合并时，如果当前区间右端点大于 t，更新 t 的右端点及其开闭性为当前区间的右端点及其开闭性；如果当前区间右端点等于 t，更新 t 的右端点的开闭性，当前区间右端点开闭性和 t 中有一者为闭则更新 t 右端点开闭性为闭，否则为开；如果当前区间右端点小于 t，则不应改变 t。

无法再进行合并时，输出 t 记录的区间合并结果，然后继续遍历二维数组，直至结束。

示例代码

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int cmp(const void *, const void *); //按区间左端点大小升序排序，当左端点相同时，闭区间在前，开区间在后。
/*
二维数组in记录输入区间
每行：
0-左端点大小
1-右端点大小
2-左端点开闭性，1表示闭，0表示开
3-右端点开闭性，1表示闭，0表示开
*/
int in[500005][4];
int main() {
    int n = 0; //记录输入的区间数量
    char l, r;
    while(~scanf("%c%d,%d%c ", &l, &in[n][0], &in[n][1], &r)) { //记得写空格跳过行末换行符
        in[n][2] = l == '[' ? 1 : 0; //巧用三目运算符，也可以用if-else
        in[n][3] = r == ']' ? 1 : 0;
        ++n;
    }
    qsort(in, n, sizeof(in[0]), cmp); //按区间左端点大小升序排序，当左端点相同时，闭区间在前，开区间在后。
    for(int i = 0; i < n;) {
        int t[4]; //记录本次合并结果
        memcpy(t, in[i], sizeof(t)); //初始化t为本次合并的最初区间
        //只有区间in[i]左端点小于t右端点，或区间in[i]左端点等于t右端点且不全为开区间，才能合并
        for(++i; i < n && (in[i][0] < t[1] || (in[i][0] == t[1] && (in[i][2] | t[3]))); ++i) { //至少一个闭时in[i][2] | t[3]为1，若均为开则为0
            if(in[i][1] > t[1]) //in[i]右端点大于t，更新t右端点及其开闭性为in[i]右端点及其开闭性
                t[1] = in[i][1], t[3] = in[i][3];
            else if(in[i][1] == t[1]) //in[i]右端点等于t，更新t右端点的开闭性（in[i]和t中有一者为闭则更新t右端点开闭性为闭，否则为开）
```

```
                t[3] |= in[i][3]; //巧用位运算
            //若in[i][1]<t[1]，则不应改变t
        }
        printf("%c%d,%d%c\n", t[2] ? '[' : '(', t[0], t[1], t[3] ? ']' : ')');
//巧用三目运算符，也可以用if-else
    }
    return 0;
}

//按区间左端点大小升序排序，当左端点相同时，闭区间在前，开区间在后。
int cmp(const void *p, const void *q) {
    int *a = (int *)p;
    int *b = (int *)q;
    if(a[0] > b[0]) return 1;
    else if(a[0] < b[0]) return -1;
    else if(a[2] < b[2]) return 1;
    else if(a[2] > b[2]) return -1;
    else return 0;
}
```

补充

示例代码中用了一些三目运算符和位运算简化了代码，同学们也可以全用 if 和 $else$ 代替。

推荐同学们去学习一下为什么二维数组排序的 cmp 函数要这么写。

本题记录区间也可以使用结构体，感兴趣的同学可以自学。

Author: 哪吒

I 猜牌游戏

题目解析

这道题本质上是个模拟题。

我们分析一下 Case 1 和 Case 2 分别在什么情况下才会说：

- 对于知道花色的空，只要该花色剩余多余一张，那么他就会选择 Case 1，否则选择 Case 2；
- 对于知道点数的荧，只要该点数剩余多余一张，那么她就会选择 Case 1，否则选择 Case 2；

换言之，如果选择了 Case 1，那么就能说明该颜色（或点数）多余一张，于是下一个人就可以将只有一张卡牌的颜色（或点数）排除。

通过排除法，一步一步消除；如果在某一轮中，两个人都无法排除任何卡牌，那么一定无法猜出来。

示例代码

```
#include<stdio.h>

int n;
char cards[13][4];
int ans_order, ans_type;
char c[3];

void get_card(char* card, int* order, int* type) {
    if (card[0] >= '2' && card[0] <= '9') *order = card[0] - '0';
    else if (card[0] == 'T') *order = 10;
    else if (card[0] == 'J') *order = 11;
    else if (card[0] == 'Q') *order = 12;
    else if (card[0] == 'K') *order = 0;
    else if (card[0] == 'A') *order = 1;
    *type = card[1] - 'W';
}

int main() {
    scanf("%d", &n);
    for (int i = 0;i < n;i++) {
        scanf("%s", c);
        get_card(c, &ans_order, &ans_type);
        cards[ans_order][ans_type] = 1;
    }
    scanf("%s", c);
    get_card(c, &ans_order, &ans_type);
    int cnt = 0;
    while (1) {
        int flg = 0;
        // Kong's turn
        cnt++;
        {
            int same_cnt = 0;
            for (int i = 0;i < 13;i++)
                if (cards[i][ans_type]) same_cnt++;
            if (same_cnt == 1) break;
            for (int i = 0;i < 4;i++) {
                same_cnt = 0;
                for (int j = 0;j < 13;j++)
                    if (cards[j][i]) same_cnt++;
                if (same_cnt == 1)
                {
                    for (int j = 0;j < 13;j++) cards[j][i] = 0;
                    flg = 1;
                }
            }
        }
```

```
        // Ying's turn
        cnt++;
        {
            int same_cnt = 0;
            for (int i = 0;i < 4;i++)
                if (cards[ans_order][i]) same_cnt++;
            if (same_cnt == 1) break;
            for (int i = 0;i < 13;i++) {
                same_cnt = 0;
                for (int j = 0;j < 4;j++)
                    if (cards[i][j]) same_cnt++;
                if (same_cnt == 1)
                {
                    for (int j = 0;j < 4;j++) cards[i][j] = 0;
                    flg = 1;
                }
            }
        }
        if (flg == 0) {
            printf("0");
            return 0;
        }
    }
    printf("%d", cnt);
}
```

J 工科元素分析-草

题目解读

大模拟，由题目描述易知，每个时刻要么敌人未附着元素，要么敌人附着了一种元素。对于每种敌人附着的元素，分类讨论并求解即可。

解题思路

首先读入几个基本数据，按题意可知，不再赘述。

我们设置几个变量来记录每一次的状态：

- 记录敌人身上当前附着什么元素 012345 对应 无草水雷火激
- 记录当前的草原核数量，因为要判断超/烈绽放
- 记录各个反应发生次数和草原核生成次数以备输出

然后就可以开始分类讨论：

- 首先，当次攻击不附着元素的情况下，显然不产生任何反应，直接结算伤害
- 其次，先判断超/烈绽放反应，因为这些反应只与草原核是否存在有关，并结算对应的爆炸伤害，更新原核数目。
- 之后，如果敌人未附着元素，敌人身上不发生任何反应，直接结算伤害

- 再考虑发生原/蔓激化反应的情况，强化伤害。

上述就是所有“不改变敌人身上元素类型”的可能性情况，下面再继续考虑“改变敌人身上元素类型”的情况。需要注意，唯一的改变攻击伤害的反应是原/蔓激化反应，已经考虑过了，因此后续发生什么反应，本次造成攻击伤害都一样。

- 首先是已存在激元素的反应，如果是水则生成草原核，无论如何都消耗激元素。
- 其次考虑绽放反应，生成新的草原核，注意这里和上一步都要判断草原核是否过多并结算爆炸伤害，清空敌人身上元素。
- 最后考虑原激化反应，生成激元素。
- 运行到最后说明发生了其他元素之间的反应，清空元素即可。

其他反应计数在对应反应发生时计数即可，草原核计数在生成时计数即可，注意生成完就爆炸的也要计数。

最后输出即可。

代码示例

```
#include <stdio.h>

const double base = 1446.853458;

// normal damage calc function
double normal_damage(double atk, double cr, double cd){
    return atk * (1 + cr * cd);
}

// burst damage
double burst(double em, double type){
    return type * base * (1 + (16 * em) / (2000 + em));
}

// intensify strengthened damage
double intensify(double atk, double cr, double cd, double em, double type){
    return normal_damage(atk + (type * base *  (1 + (5 * em) / (1200 + em))),
cr, cd);
}

int main()
{
    int N;
    double atk, cr, cd, em;
    // atk critical_rate critical_damage ele_mastery
    scanf("%d%lf%lf%lf%lf", &N, &atk, &cr, &cd, &em);
    // 0=none 1=grass 2=water 3=thunder 4=flame 5=intensify
    int cur_ele = 0, atk_ele;
    // how much grass core remain
```

```
    int core_cnt = 0;
    // thun=thunder gra=grass in=intensify bur=burst
    int thun_in = 0, gra_in = 0, core_sum = 0, thun_bur = 0, flame_bur = 0;
    double sum_dam = 0;
    while (N--){
        scanf("%d", &atk_ele);
        // attack with no element
        if (atk_ele == 0){
            sum_dam += normal_damage(atk, cr, cd);
            continue;
        }
        // flame/super burst
        if (core_cnt && (atk_ele == 3 || atk_ele == 4)){
            sum_dam += core_cnt * burst(em, 3);
            if (atk_ele == 3)thun_bur += core_cnt;
            else flame_bur += core_cnt;
            // clear grass core count
            core_cnt = 0;
        }
        // enemy has no element
        if (cur_ele == 0){
            cur_ele = atk_ele;
            sum_dam += normal_damage(atk, cr, cd);
            continue;
        }
        // grass/super intensify
        if (cur_ele == 5 && (atk_ele == 1 || atk_ele == 3)){
            sum_dam += intensify(atk, cr, cd, em,
                    atk_ele == 1 ? 1.15 : 1.25);
            if (atk_ele == 3)thun_in++;
            else gra_in++;
            continue;
        }

        // latter reactions have same attack damage
        sum_dam += normal_damage(atk, cr, cd);

        // intensify element & water element cause burst
        if (cur_ele == 5 && atk_ele == 2){
            core_sum++;
            if (core_cnt == 5){
                sum_dam += burst(em, 2);
            }
            else {
                core_cnt++;
            }
            cur_ele = 0;
            continue;
        }
        // intensify element is reacted
```

```
        if (cur_ele == 5){
            cur_ele = 0;
            continue;
        }
        // same element no reaction
        if (cur_ele == atk_ele){
            continue;
        }
        // water & grass cause burst
        if ((cur_ele == 1 && atk_ele == 2) || (cur_ele == 2 && atk_ele == 1)){
            core_sum++;
            if (core_cnt == 5){
                sum_dam += burst(em, 2);
            }
            else {
                core_cnt++;
            }
            cur_ele = 0;
            continue;
        }
        // thunuder & grass cause intensify
        if ((cur_ele == 1 && atk_ele == 3) || (cur_ele == 3 && atk_ele == 1)){
            cur_ele = 5;
            continue;
        }
        // other reaction clear the element
        cur_ele = 0;
    }
    sum_dam += core_cnt * burst(em, 2);
    printf("%.0lf %d %d %d %d %d", sum_dam, thun_in, gra_in, core_sum,
thun_bur, flame_bur);
    return 0;
}
```