C8 题解

A 告别的字符画

考点	难度
转义字符	1

题意分析

挥挥手！处理一下转义字符！

在 `c` 语言的字符串中需要转换的转义字符有：

输出 `"` 需要使用 `\"`

输出 `'` 需要使用 `\'`

输出 `\` 需要使用 `\\`

另外，如果你使用 `printf` 输出的话，额外有一个转换:

输出 `%` 需要使用 `%%`

虽然本题手动修改很简单，可如果是数据量较大的字符画，善用**搜索-替换**功能，按 `ctrl+F` 一般就能打开搜索-替换界面，搜索对应的需要处理的字符并转义即可。

注意要先处理 `\` 不然已经处理过后的转义字符会被再处理一次。

或者，你可以尝试写一个 `一劳永逸解决字符画.c` ，比如

```c
#include <stdio.h>
char c;
int main(void) {
    freopen("AA.c", "w", stdout);
    puts("#include <stdio.h>");
    puts("int main()");
    puts("{");
    while (~(c = getchar())) {
        printf("    puts(\"");
        for (; c != '\n'; c = getchar()) {
            if (c == 34 || c == 39 || c == 92)
                putchar(92);
            putchar(c);
        }
        puts("\");");
    }
    puts("  return 0;");
    puts("}");
    return 0;
}
```

编译并运行这个程序，然后输入你需要处理的字符画，这个程序就会在当前目录下生成一个 `AA.C` ，这个就是输出字符画的程序，你可以随意编辑，复制等。

示例代码

```
#include <stdio.h>
int main() {
    puts("~~~\\\"\'(^.^ )\'\"/~~~ByeBye!!");
    return 0;
}
```

B HiDen的大一下回忆

这题考察大家对数学运算里向上取整和向下取整的运用以及生活中实际情况的判断。第一次进校，一定有m个人能进去，后面最多带$m-1$个人进校，来回一共两个单位的时间。如果只有一张卡，且总人数多于两人，那么无论如何所有人都不可能进校。

参考代码

```
#include<stdio.h>
int main()
{
    int t,m,n;
    scanf("%d",&t);
    while(t--){
        scanf("%d%d",&n,&m);
        if(m==1&n!=1) printf("-1\n");//特判，只有一张卡两个人及以上是不行的。
        else{
            if(m==1&&n==1) printf("1\n");
            else{
                int k = (n-m)/(m-1);
                int tmp = (n-m)%(m-1);
                if(tmp==0) printf("%d\n",k*2+1);//tmp等于0说明每次进去的人数都是m。
                else printf("%d\n",(k+1)*2+1);//最后一次要运送少于m个的人数。
            }
        }
    }
    return 0;
}
```

C 唐朝诡事录之逃出密室

考点	难度
位运算	2

题目分析

本题考察简单位运算，在Hint的提示下，解题思路非常明确。

推荐大家记住本次Hint的内容：

1. 将十进制数 num 的二进制表达形式（最低位编号为 0 ）的第 i 位置0：`num&(~(1<<i))`；
2. 将十进制数 num 的二进制表达形式（最低位编号为 0 ）的第 i 位置1：`num|(1<<i)`；
3. 将十进制数 num 的二进制表达形式（最低位编号为 0 ）的第 i 位翻转：`num^(1<<i)`；

示例代码

```
#include<stdio.h>

int num,i,op,high,low;
int main()
{
    scanf("%d",&num);
    while(scanf("%d%d%d",&op,&high,&low)!=EOF)
    {
        if(op==0)
        {
            for(i=low;i<=high;i++)
            {
                num=num&(~(1<<i));
            }
        }
        else if(op==1)
        {
            for(i=low;i<=high;i++)
            {
                num=num|(1<<i);
            }
        }
        else
        {
            for(i=low;i<=high;i++)
            {
                num=num^(1<<i);
            }
        }
    }
    printf("%d",num);
    return 0;
}
```

D ljh的成绩分析ljh的成绩分析

难度	考点
2	排序

问题分析

题目要求比较明确，即三个任务：排序、平均值、满分人数。考虑数据范围，此处使用冒泡排序就可以了，计算平均值相信大家都很熟悉了，至于统计满分人数，大家既可以在输入的同时用`if`语句进行判断，也可以在排完序后从右往左枚举。最后注意一下输出格式就OK了。

参考代码

```
#include <stdio.h>
void Bubble_sort(int a[], int n) //冒泡排序
{
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n - i - 1; j++)
        {
            if (a[j] >= a[j + 1])
            {
                int temp = a[j];
                a[j] = a[j + 1];
                a[j + 1] = temp;
            }
        }
}
int main()
{
    int n, a[5010] = {0};
    double avg = 0;
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &a[i]);
        avg += a[i]; //计算累加和以求平均值
    }
    Bubble_sort(a, n);
    int cntak = 0;
    for (int i = n - 1; i >= 0 && a[i] == 110; i--) //统计满分人数，注意循环结束条件
        cntak++;
    for (int i = 0; i < n; i++) //输出排序完成后的序列
        printf("%d ", a[i]);
    printf("\n%.2lf\n", avg / n); //输出平均值
    if (!cntak) //不存在ak的
        printf("It's a great pity that no one got a perfect score.\n");
    else //存在，输出人数
        printf("There are %d people who got full marks!\n", cntak);
}
```

当然，用`qsort`快速排序也是可以的：

```
#include <stdio.h>
#include <stdlib.h>
int cmp(const void *a,const void *b)
{
    int A=(*(int *)a),B=(*(int *)b);
    if(A<B) return -1;
    if(A>B) return 1;
```

```
    return 0;
}
int main()
{
    int n, a[5010] = {0};
    double avg = 0;
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &a[i]);
        avg += a[i];
    }
    qsort(a,n,sizeof(int),cmp);
    int cntak = 0;
    for (int i = n - 1; i >= 0 && a[i] == 110; i--)
        cntak++;
    for (int i = 0; i < n; i++)
        printf("%d ", a[i]);
    printf("\n%.2lf\n", avg / n);
    if (!cntak)
        printf("It's a great pity that no one got a perfect score.\n");
    else
        printf("There are %d people who got full marks!\n", cntak);
}
```

E void排昵称

题目分析

本题为考察qsort的基础题，同时数据规模也允许使用冒泡排序等O(len*n^2)复杂度的排序算法通过

在此强调一下qsort函数的用法

```
qsort(a,n,sizeof(a[0]),cmp);
//qsort有四个参数
//第一个为排序的首地址，特别的，此处这个用法指的是从a[0]开始排序，对a[0]...a[n-1]进行排序，
//如果想要对a[1]...a[n]进行排序，那么第一个参数就要写：a+1
//第二个为需要排序的元素个数
//第三个为单个元素所占用的字节大小
//第四个为比较规则，用两个输入的void型指针，以返回值的形式给出哪个应当排在前面
```

此外，对于一个char型的二维数组而言，它也可以被视作一个以字符串为元素的数组，每一行为一个字符串，可以使用scanf，gets等读入

代码示例

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<ctype.h>
#include<string.h>
#include<time.h>
#define REP(i,a,b) for(register int i=a;i<=b;i++)
#define PER(i,a,b) for(register int i=a;i>=b;i--)
```

```
typedef long long LL;
char s[1010][110];
int n=1;
int cmp(const void *a,const void *b){
    return strcmp((char*)a,(char*)b);//指针的强制类型转换
}
int main(){
    while(scanf("%s",&s[n])!=EOF)n++;
    n--;
    qsort(s+1,n,sizeof(s[1]),cmp);
    //或者可以写:qsort(s+1,n,sizeof(s[1]),strcmp);
    //能这样写的原因是，strcmp刚好能对两个输入的指向字符串的指针给出一个字典序的比较结果
    //其中应该涉及到参数的类型转换
    REP(i,1,n)puts(s[i]);
}
```

F 幸运日

题目分析

本题需要判断在 $[D1, D2]$ 范围内的日期是否包含“102”。如果想要直接枚举日期会比较麻烦，我们可以枚举 $[D1, D2]$ 范围内的所有数字，判断其是否符合日期规范，然后再判断其是否包含“102”。

1. 判断日期合法性：对于一个八位数字 D，取出 $year, month, day$，分别判断，并注意对闰年的处理。
2. 判断是否包含“102”：可以将 `date` 转换为字符串，当 `strstr(s, "102") != NULL` 时说明存在子串 102102。另外，将 `int` 型转换为 `char` 型有库函数 `atoi(str)`，但oj并不支持。在这里我们提供另一种便捷的做法：`sprintf(str, "%d", num) //将num转为字符串输入到str中`

示例代码

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

int daynum[] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}; //预存每个月天数

int checkDate(int date) {
    int day = date % 100, month = (date / 100) % 100, year = date / 10000; //取出year，month，day
    if (year % 100 != 0 && year % 4 == 0 || year % 400 == 0) {//判断是否为闰年，若是，则2月应为29日，否则为28日
        daynum[2] = 29;
    } else {
        daynum[2] = 28;
    }
    if (1 <= month && month <= 12 && 1 <= day && day <= daynum[month]) { // 判断日期合法性
        return 1;
    }
    return 0;
}

int main() {
```

```
    char single[10];
    int start, end;
    while (~scanf("%d%d", &start, &end)) {
        int cnt = 0; // cnt存储符合条件的日期数量
        for (int i = start; i <= end; ++i) {
            if (checkDate(i)) { //判断日期合法性
                sprintf(single, "%d", i); //转为字符串
                if (strstr(single, "102")) { //判断是否包含“102”
                    ++cnt;
                }
            }
        }
        printf("%d\n", cnt);
    }

    return 0;
}
```

谢谢来自 *Asahi* 提供的代码 (*^▽^*)

Addition

另外，如果本题数据加强，比如不保证 $|D2 - D1| \leq 100000$, 数据组数更多的情况下，暴力枚举将会 tle，这里提供另一种思路，供学有余力的同学学习。

通过思考我们可以发现对于八位日期可能存在的包含 “102” 的情况有以下 5 种：

$$①102xxxxx$$
$$②x102xxxx$$
$$③xxx102xx$$
$$④xxxx102x$$
$$⑤xxxxx102$$

对于情况①②年份中包含“102”，则该年份所有日期均为幸运日。

对于情况③，该年份2月份均为幸运日。

对于情况④，10月2x日均为幸运日。

对于情况⑤，有 xxxx0102 和 xxxx1102 两种情况。

因此对于在 D1 和 D2 间的年份（不包含D1, D2 所属的年份）我们可以直接进行统计，对于 D1、D2 所属的年份且在区间内的日期进行暴力枚举。

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int l, r;
int month[15] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
int isLeapYear(int year) { // 判断是否为闰年
    return year % 400 == 0 ? 1 : ((year % 4 == 0 && year % 100 != 0) ? 1 : 0);
}
int islucky(int date) { // 判断是否包含“102”
    char r[10];
    sprintf(r, "%d", date);
    return strstr(r, "102") == NULL ? 0 : 1;
```

```
}
int check(int date) { // 判断是否为合法日期
    int y = date / 10000;
    int m = (date % 10000) / 100;
    int d = date % 100;
    month[2] = 28 + isLeapYear(y);
    if (m < 1 || m > 12) return 0;
    if (d < 1 || d > month[m]) return 0;
    return 1;
}
int main()
{
    int y1, y2, ans;
    while (scanf("%d%d", &l, &r) != EOF) {
        ans = 0;
        y1 = l / 10000; //取year
        y2 = r / 10000;
        for (int i = y1 + 1; i < y2; i++) { //对区间内的年份（不包含边界）直接进行统计
            if (i / 10 == 102 || i % 1000 == 102) // 情况1,2
                ans += 365 + isLeapYear(i);
            else if (i % 10 == 1) // xxx1: 包含情况3,4,5
                ans += 28 + isLeapYear(i) + 10 + 2;
            else // xxxx: 包含情况4,5
                ans += 10 + 2;
        }
        if (y1 == y2) { // 如果D1,D2在同一年
            for (int i = l; i <= r; i++)
                if (check(i))
                    ans += islucky(i);
        }
        else {//如果D1,D2不在同一年
            for (int i = l; i <= (l / 10000 + 1) * 10000; i++) // D1->D1所在年的最
后一天
                if (check(i))
                    ans += islucky(i);
            for (int i = r / 10000 * 10000; i <= r; i++) // D2所在年的第一天->D2
                if (check(i))
                    ans += islucky(i);
        }
        printf("%d\n", ans);
    }
    return 0;
}
```

G 保护大结界

难度	考点
2	快排

问题分析

考虑下述函数：

$$\mathrm{f}(x)=|x-a|+|x-b| \quad (a\le b)$$

其函数值有如下情况：

$$\mathrm{f}(x)=\begin{cases} a+b-2x\,, & x<a \\ b-a\,, & a\le x<b \\ 2x-a-b\,, & x\ge b \end{cases}$$

易知当 $x<a$ 或 $x\ge b$ 时，$\mathrm{f}(x)\ge b-a$，即当 $a\le x\le b$ 时取得最小值。（提示：可以在数轴上标点画图来快速得到结果）

事实上，由上面可以得到，对于函数：

$$\mathrm{F}(x)=\sum_{i=1}^{n}|x-a_i| \quad (a_i\le a_{i+1})$$

当 $n=2k-1(k\in\mathbb{N}^*)$ 时，$\mathrm{F}(x)$ 在 $x=a_{\frac{n+1}{2}}$ 处取得最小值；

当 $n=2k(k\in\mathbb{N}^*)$ 时，$\mathrm{F}(x)$ 在 $a_{\frac{n}{2}}\le x\le a_{\frac{n}{2}+1}$ 时取得最小值。

由于证明类似，此处略去。

由上述证明可知，当 x 取 $\{a_n\}$ 的中位数时，$\mathrm{F}(x)$ 取得最小值。

注意到 x,y,z 相互独立，因此可以分别排序求解。

参考代码

```c
#include <stdio.h>
#include <stdlib.h>

int mycomp(const void *p1, const void *p2)
{
    const int *a1 = (const int *)p1;
    const int *a2 = (const int *)p2;
    if (*a1 < *a2)
        return -1;
    else if (*a1 == *a2)
        return 0;
    else
        return 1;
}

int main()
{
    int n, d[3][100005];
    long long sum = 0;
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
        for (int j = 0; j < 3; j++)
            scanf("%d", &d[j][i]);
    for (int i = 0; i < 3; i++)
    {
```

```
        qsort(d[i], n, sizeof(int), mycomp);
        for (int j = 0; j < n / 2; j++)
            sum += d[i][n - j - 1] - d[i][j];
    }
    printf("%lld", sum);
    return 0;
}
```

H Red的数独谜题

题目分析

这道题目的题意是给出一个数独，要求判断这个数独是否合法，若不合法则输出两个数字的位置。

首先我们按规则判定其是否合法。因只交换了两数字，则必有某行或某列不合法，故可以不考虑宫的合法性。

若合法则可直接输出。若不合法，为了确定不合法的位置，我们可以先找到不合法的行和列，然后通过行和列的位置确定不合法的数字。若仅有行或列不合法，则为同一列/行的两个数字，可以依次判断每列/行不合法的数字数量来确定该行/列。

示例代码

```
#include <stdio.h>

void isLegal(int a[][9])
{
    int flagR = 0, flagC = 0, errorrow[9][9] = {}, errorcol[9][9] = {};
    for (int i = 0; i < 9; i++)
    {
        int row[10] = {}, col[10] = {}; // 用于判断第i行/列是否有重复数字
        for (int j = 0; j < 9; j++)
        {
            if (row[a[i][j]]) // 第i行有重复数字，则记录于errorrow中，同时记录存在错误的
行
                errorrow[i][a[i][j]] = 1, flagR++;
            if (col[a[j][i]]) // 同上
                errorcol[i][a[j][i]] = 1, flagC++;
            row[a[i][j]]++, col[a[j][i]]++; // 记录第i行/列中出现了的数字
        }
    }
    if (flagR && flagC) // 若行列均有错误，则说明交换了不同行列的两个数
    {
        for (int i = 0; i < 9; i++)
            for (int j = 0; j < 9; j++)
                if (errorrow[i][a[i][j]] && errorcol[j][a[i][j]]) // 若在行和列
                    printf("(%d,%d) ", i + 1, j + 1);
    }
    else if (flagR || flagC) // 若只有行或列有错误，则说明交换了同一行列的两个数
    {
        for (int i = 0; i < 9; i++)
        {
            int cnt = 0;
            for (int j = 0; j < 9; j++) // 统计每列/行中错误的数字的个数
                cnt += flagR ? errorrow[j][a[j][i]] : errorcol[j][a[i][j]];
```

```
            if (cnt == 2) // 若错误的数字个数为2，则说明交换了同一行/列的两个数
            {
                for (int j = 0; j < 9; j++)
                {
                    if (flagR && errorrow[j][a[j][i]])
                        printf("(%d,%d) ", j + 1, i + 1);
                    else if (flagC && errorcol[j][a[i][j]])
                        printf("(%d,%d) ", i + 1, j + 1);
                }
            }
        }
    }
    else
        printf("Congratulations!");
}

int main()
{
    int a[9][9];
    for (int i = 0; i < 9; i++)
        for (int j = 0; j < 9; j++)
            scanf("%d", &a[i][j]);
    isLegal(a);
    return 0;
}
```

I 求求你了帮帮可莉吧1

当整数过大时，`longlong`类型都不足以支持我们的运算了，这时我们就要考虑原始的方式去进行计算。

本题考察大整数的乘法，只需用上我们的小学数学知识，竖式运算，模拟算乘法的过程，一步一步来。

比如：

```
    123
×    21
---------
    123
   246
---------
   2583
```

存储大整数，我们就用字符串的形式存入，那么`s[0]`就是最高位的数字，`s[1]`就是次高位。再将每一位转换为整数的形式放入`int`类型数组中，接下来开始按照乘法的方式用一个乘数的每一位去乘另一个乘数的所有位，这里我们其实不用考虑进位，可以等到最后再处理，因为每个数组存的是`int`类型，并不是一个数字。

参考代码：

```
#include<stdio.h>
#include<string.h>
char s1[1005],s2[1005];//存放两个大整数的字符
int int1[1005],int2[1005];//按低位到高位存放两个大整数的数组
```

```
int result[1000005];//乘积结果数组 ,按低位到高位存放
int main() {
    int i,j;//循环变量
    int d;//进位
    int b;//存放按位乘的结果
    int m;//结果数组的下标
    int len_result;
    //获取两个大整数
    scanf("%s%s",s1,s2);
    //按低位到高位存放两个大整数
    int len1 = strlen(s1);
    int len2 = strlen(s2);
    for(i=0,j=len1-1; i < len1; i++,j--) {
        int1[j]=s1[i]-'0';
    }

    for(i=0,j=len2-1; i<len2; i++,j--) {
        int2[j]=s2[i]-'0';
    }

    //char字符数组长度等于对应的int字符数组长度
    //模拟竖式相乘
    for(i=0; i<len2; i++) {
        d=0;//一开始相乘,进位为0
        m=i;
        for(j=0; j<len1; j++) {
            b=int2[i]*int1[j]+d+result[m];
            result[m]=b%10;//本位
            m++;//为下一次做准备
            d=b/10;//进位
        }

        if(d>0) { //一次相乘，最后一次的进位
            result[m]=d;
        }
    }
    //输出结果
    len_result= len1 + len2;
    while(result[len_result]==0&&len_result>0) { // 删除前导符0
        len_result--;
    }
    for(i=len_result; i>=0; i--) {
        printf("%d",result[i]);
    }
    return 0;
}
```

J 指尖宇宙

题目解析

这道题是一道有关字符串的模拟题，整体难度不高，但需要注意以下几点：

1. 数据范围保证宇宙面积上限，但不保证宽和高，所以开二维的静态数组是肯定不行的。解决方案：使用 `malloc` 动态申请内存或使用一维数组表征二维。
2. 这些字符串读入有空格，所以不能用 `scanf`，而应该用 `gets`。随之而来的问题就是 `gets` 使用前需要忽略前面的换行符，于是使用 `scanf` 的时候需要在格式串结尾添加空格忽略掉空白字符。

宇宙可以看作一个字符矩阵（仅有可见字符与空格构成）。初始状态下，宇宙有 m 行 n 列。$\mathcal{Jery}$ 将对宇宙进行 T 次操作，他想知道这 T 次操作之后的宇宙长什么样子。

具体来说，操作有以下几种：

指尖膨胀

指定一个**可见字符**，将宇宙中所有该字符替换成一个 m_0 行 n_0 列的矩形，同时，其它位置都应该等比放大，且用该位置原来的字符填充该矩形。

注意，进行该操作后，矩阵的行数会变为原来的 m_0 倍，列数会变成原来的 n_0 倍。（即使宇宙中不存在指定的字符，宇宙每个位置也会按该比例放大）

举个例子，假如原来的宇宙为一个 3 行 5 列的字符矩阵：

```
## 01
#0.&&
0   #
```

指定字符 `#` 并将其替换为一个 2 行 4 列的字符矩阵：

```
aA B
C K0
```

替换后的结果为一个 6 行 20 列的字符矩阵：

```
aA BaA B    00001111
C K0C K0    00001111
aA B0000....&&&&&&&&
C K00000....&&&&&&&&
0000            aA B
0000            C K0
```

解决方案

新开一个宇宙，按一定比例复制过去。

指尖湮灭

指定一行（列），将该行（列）消除，上下（左右）合并。

举个例子，假如原来的宇宙为一个 3 行 5 列的字符矩阵：

```
## 01
#0.&&
0   #
```

指定第 4 列消除，则结果为一个 3 行 4 列的字符矩阵：

```
## 1
#0.&
0  #
```

解决方案

新开一个宇宙，判断删行还是删列，并把后面的行（列）向前平移即可。

指尖坍缩

指定一个 r 行 c 列的字符矩阵作为坍缩因子，同时指定一个字符 ord 作为秩序符号，以及一个字符 mes 作为混沌符号。设宇宙为 M 行 N 列，这里保证 r 是 M 的因子，c 是 N 的因子。

将宇宙按 r 行 c 列为一个单位分成若干块，坍缩后，每一个单位块均会变成一个字符，于是宇宙变成 $\frac{M}{r}$ 行 $\frac{N}{c}$ 列。

对于每一个单位块，如果它与坍缩因子相同（这里的相同指这 $r \times c$ 个字符逐位置比较均相同），那么将它变成秩序符号 ord。否则，判断该单位块中非空格的字符类型数 d：

- 如果 $d = 0$，即该单位块全都是空格，那么将该单位块变为一个空格；
- 如果 $d = 1$，即除空格之外有且仅有一种字符，那么将该单位块变为该字符；
- 如果 $d \geq 2$，即除空格之外有至少 2 种字符，那么将该单位块变为混沌符号 mes。

举个例子，假如原来的宇宙为一个 2 行 6 列的字符矩阵：

```
##0  #
#00. #
```

指定字符 `P` 和 `@` 分别作为秩序符号 ord 和混沌符号 mes。并指定坍缩因子为一个 2 行 1 列的字符矩阵：

```
#
#
```

替换后的结果为一个 1 行 6 列的字符矩阵：

```
P@0. P
```

解决方案

新开一个宇宙，逐块判断坍缩类型，并赋值即可。

示例代码

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

char tmp[3][3000000];
char* universe = tmp[0], * mid = tmp[1], * dst = tmp[2];
int m, n;
```

```
void input_matrix(char* mat, int row, int column) {
    for (int i = 0;i < row;i++) gets(mat + (column * i));
}

void swap_matrix(char** mat1, char** mat2) {
    char* tmp = *mat1;
    *mat1 = *mat2;
    *mat2 = tmp;
}

void handle() {
    char cmd[100];
    gets(cmd);
    if (*cmd == '1') {
        int m0, n0;
        scanf("%d%d ", &m0, &n0);
        input_matrix(mid, m0, n0);
        for (int i = 0;i < m;i++)
            for (int j = 0;j < n;j++)
                for (int r = 0;r < m0;r++)
                    for (int c = 0;c < n0;c++)
                        dst[(i * m0 + r) * n0 * n + (j * n0 + c)] =
                            ((universe[i * n + j] == cmd[1])
                             ? mid[r * n0 + c]
                             : universe[i * n + j]);
        swap_matrix(&dst, &universe);
        m *= m0;
        n *= n0;
    }
    else if (*cmd == '2') {
        int index;
        scanf("%d ", &index);
        if (cmd[1] == 'r') {
            for (int i = index - 1;i < m - 1;i++)
                for (int j = 0;j < n;j++)
                    universe[i * n + j] = universe[(i + 1) * n + j];
            m -= 1;
        }
        else if (cmd[1] == 'c') {
            for (int i = 0;i < m;i++)
                for (int j = 0;j < n - 1;j++)
                    dst[i * (n - 1) + j] =
                        ((j < index - 1) ? universe[i * n + j] : universe[i * n
+ j + 1]);
            n -= 1;
            swap_matrix(&dst, &universe);
        }
    }
    else if (*cmd == '3') {
        int r, c;
        scanf("%d%d ", &r, &c);
        input_matrix(mid, r, c);
        int m0 = m / r, n0 = n / c;
#define sr(_r, _c) universe[(_r)*n+(_c)]
#define ds(_r, _c) dst[(_r)*n0+(_c)]
```

```
#define md(_r, _c) mid[(_r)*c+(_c)]
        for (int i = 0;i < m0;i++)
            for (int j = 0;j < n0;j++) {
                int apr[1000] = { 0 };
                int d = 0;
                int same = 1;
                char last;
                for (int r0 = 0;r0 < r;r0++)
                    for (int c0 = 0;c0 < c;c0++) {
                        if (sr(i * r + r0, j * c + c0) != md(r0, c0)) same = 0;
                        if (sr(i * r + r0, j * c + c0) == ' ') continue;
                        if (apr[sr(i * r + r0, j * c + c0)]) continue;
                        apr[sr(i * r + r0, j * c + c0)] = 1;
                        d += 1;
                        last = sr(i * r + r0, j * c + c0);
                    }
                if (same) ds(i, j) = cmd[1];
                else if (d == 0) ds(i, j) = ' ';
                else if (d == 1) ds(i, j) = last;
                else ds(i, j) = cmd[2];
            }
        swap_matrix(&dst, &universe);
        m = m0;
        n = n0;
    }
    else printf("ERROR: %c %d", *cmd, *cmd);
}

int main() {
    int t;
    scanf("%d%d%d ", &m, &n, &t);
    input_matrix(universe, m, n);
    // printf("%s", universe);
    while (t--) {
        handle();
    }
    for (int i = 0;i < m;i++) {
        for (int j = 0;j < n;j++) printf("%c", universe[i * n + j]);
        printf("\n");
    }
}
```