2014 级 C++第六次练习解题报告

14211067 朱福

(A)jhljx 又来学函数

Problem Description

jhljx 最近很爱学习吖。。咕~~(ノ ﹏ ヽ)b。。他决定把函数重新温习一下。。233
于是他找到了这样一个函数。

$$A(m,n)=\begin{cases} n+1 & m=0 \\ A(m-1,1) & m>0,n=0 \\ A(m-1,A(m,n-1)) & m>0,n>0 \end{cases}$$

这个函数好神奇吖。。他给了你一个 m 和 n,让你来求出 A(m,n)的值。

Input

输入多组数据。
每组数据一行。包括两个数 m 和 n。m 可以取 0,1,2,3。当 m 取 0,1,2 时，0<=n<=1000000。当 m 取 3 时，0<=n<=24。

Output

每组数据输出一行，为一个数，即 A(m,n)。

Sample Input

```
1 3
2 4
```

Sample Output

5
11

Hint

/*这是你们看不到的注释
这道题可以用递归来水。但是得不到全分哦。。要想拿满分，去百度一下把。。
啦啦啦~~~
我不会说网上会有计算公式哦。。

*/

解析：本题利用数列递推来做。
m=0 时 A(0,n)=n+1
m=1 时 A(1,n)=A(0,A(1,n-1))=A(1,n-1)+1，由递推知 A(1,n)=n+2
m=2 时 A(2,n)=A(1,A(2,n-1))=A(2,n-1)+2，由递推知 A(2,n)=A(2,0)+2*n
而 A(2,0)=A(1,1)=3，故 A(2,n)=2*n+3
m=3 时 A(3,n)=A(2,A(3,n-1))=2*A(3,n-1)+3，可以推出 A(3,n)=2^(n+3)-3

易错点：如果用循环的话会超时。

参考代码：

```
#include<iostream>
using namespace std;
int main()
{
    long long m, n, i, r;
    while (cin >> m >> n)
    {
        switch (m)
        {//对m进行分情况讨论
        case 0:
            r = n + 1;
            break;
        case 1:
            r = n + 2;
            break;
        case 2:
            r = 2 * n + 3;
            break;
        case 3:
            r = 1;
            for (i = 1; i<n + 4; i++)r *= 2;
            r -= 3;
```

```
            break;
        }
        cout << r << endl;
    }
    return 0;
}
```

(B)Results

Description

宋友老师给了 Thor 一些同学的成绩，让他把柱状图打出来。现在他来拉你做壮丁了。
具体请看下面：的样例。

Input

多组数据。
每组数据有 2 行。
第一行有 1 个整数 n(1<=n<=1000)。
第二行有 n 个整数 A[i](0<=A[i]<=100)表示每个人的成绩。

Output

对于每组数据
输出柱状图，用'*'输出。
每次输出场宽为 4。

Sample Input

```
4
80 80 90 1
5
60 65 64 63 62
```

Sample Output

```
      *
  *   *   *
  1  80  90

  *   *   *   *   *
 60  62  63  64  65
```

Hint

注意每组输出最后有一个换行。

解析：此题使用 for 循环来输出柱状图，注意什么时候输出星号，什么时候输出空格。
易错点：每组数据之间需要输出一个换行。

参考程序：

```
#include<iostream>
#include<iomanip>
using namespace std;
int main()
{
    int n, t, i;
    while (cin >> n)
    {
        int A[101] = { 0 },mx = -1;
        for (i = 0; i<n; i++)
        {
            cin >> t; A[t]++;//统计分数为t的人数
        }

        for (i = 0; i<101; i++)
        {
            if (A[i]>mx)mx = A[i];//寻找人数最多的分数点
        }

        for (int j = mx; j>0; j--)
        {
            for (i = 0; i<101; i++)
            {
                if (A[i] == 0)continue;
```

```
                if (A[i] >= j)cout << "     *"; //输出星号
                else cout << "       ";
            }
            cout << endl;
        }

        for (i = 0; i<101; i++)
        {
            if (A[i])cout << setw(4) << i;//场宽设为4
        }

        cout << endl << endl;
    }
}
```

(C)The Super N

题目描述

马上就要放假了，Alice 和 Bob 觉得非常无聊，所以决定一起玩一个游戏。由于 Alice 和 Bob 都不是特别聪明，所以游戏规则非常简单：

Alice 给出一个字母 C（小写'a'-'z'）和一个数字 n，而 Bob 则要以这个字母为起始，按招 Alice 的要求画出一个大小为 n 的字母'N'。

但是 Alice 没有告诉 Bob 他规则的具体内容，只给了 Bob 几个例子：

比如以'a'为起始字母，大小为 3 的字母 N 张这个样子：

```
a e
bdf
c g
```

再比如以'z'为起始字母，大小为 5 的字母 N 张这个样子：

```
z   h
ae  i
b f j
c  gk
d   l
```

Bob 看了之后非常费解，完全不知道到底要怎么画这个‘N‘.但是游戏已经开始了，所以，Bob 就只能求助于智商更高的你了。

如果你能帮助 Bob 解决这个问题，Alice 还会赐予你‘The Super N’的称号！

输入格式

第一行一个正整数 T(T<=100)，表示测试数据组数。

每组测试数据一行,每行一个小写字母 C('a'-'z')和一个数字 n(3<=n<=10)

输出格式

每组测试数据输出 n 行(因为要输出的字母‘N’是 n 行~跟 Alice 关系很好的 NExPlain 这么提示着你)，按题目要求输出。

样例输入

```
2
a 3
q 5
```

样例输出

```
a  e
bdf
c g
q   y
rv  z
s w a
t  xb
u   c
```

Source

BUAA_NExPlain

解析：这一题是 26 个字母循环输出的问题，可以将 char 型数据转成 int 型数据，通过对 26 取模来实现循环。
易错点：注意观察字母序号依次递增时，其位置的变化，是先从第一列开始，再是对角线，接着是最后一列。

参考程序：

```
#include<iostream>

using namespace std;

int main()
{
    int t, n,i, j;
    char c;

    cin >> t;
    while (t--)
    {
        cin >> c >> n;
        for (i = 1; i <= n; i++)
            for (j = 1; j <= n; j++)
            {
                if (j == 1)//考虑第一列
                    cout << (char)('a' + ((c + i - 1 - 'a') % 26));
                //第i个字母

                else if (j == n)//考虑最后一列
                    cout << (char)('a' + ((c + 2 * n - 3 + i - 'a') % 26)) << endl;
                //第2*n-2+i个字母

                else if (i == j)//考虑对角线
                    cout << (char)('a' + ((c + n - 2 + i - 'a') % 26));
                //第n+i-1个字母

                else cout << " ";
            }
    }
    return 0;
}
```

(D)盗墓笔记之鬼玺

题目描述：

看密匝匝蚁排兵,乱纷纷蜂酿蜜,急攘攘蝇争血。天真刚刚躲过怪物的洗礼，便赶上了这千年不遇的鬼俑阵列。
我们的故事在这之前。
少顷前，Thor 拿着鬼玺也遭遇了鬼俑的阵列，虽然拿着鬼玺，然而要号令鬼俑，还需要吸日月之灵气，汲日月之精华，按照要求将鬼俑排列好。
小哥告诉 Thor，阵列为正方形，其中鬼俑只有两种，普通步兵俑，执旗步兵俑。
对于一个 n*n 的阵列,要求每一行,每一列以及每一斜行都只有一个执旗步兵俑。
(n 个执旗步兵俑必须要放入！！)

输入：

多组测试数据。
对于每组测试数据，第一行为一个整数 n（1≤n≤20），表示阵列的边长。
接下来 n 行为一个 n*n 的矩阵，表示 Thor 排列出的阵列。其中 0 为普通步兵俑，1 为执旗步兵俑。

输出：

对于每组数据，如果列阵成功，则输出 Move Forward!
反之输出 Bad End!

输入样例：

```
8
10000000
00001000
00000001
00000100
00100000
00000010
01000000
00010000
```

输出样例：

Move Forward!

解析：本题类似于八皇后问题，但要简单一些，要保证每行每列每斜行的执旗步兵俑数不超过 1。
易错点：考虑斜行要考虑两个倾斜方向，不可只考虑一个。另外，A[i-j]（见下方代码）会造成数组标号为负数，因而要加上一个固定的数字。

参考程序：

```
#include<iostream>

using namespace std;

int main()
{
    int n, i;
    char t;
    while (cin >> n)
    {
        int A[20] = { 0 }, //记录每行执旗步兵俑个数
            B[20] = { 0 }, //记录每列执旗步兵俑个数
            C[40] = { 0 },//记录每斜行执旗步兵俑个数
            D[80] = { 0 };//记录每斜行执旗步兵俑个数

        for (i = 0; i<n; i++)
            for (int j = 0; j<n; j++)
            {
                cin >> t;
                t -= '0';
                A[i] += t;
                B[j] += t;
                C[i + j] += t;
                D[i - j + n - 1] += t;
            }

        int bo = 1;
        for (i = 0; i<n; i++)
            if (A[i] * B[i] != 1)
                bo = 0;
        //每行每列执旗步兵俑只有1个

        for (i = 0; i<2 * n - 1; i++)
            if (C[i] * D[i]>1)
                bo = 0;
        //每斜行执旗步兵俑至多1个
```

```
        if (bo)cout << "Move Forward!\n";
        else cout << "Bad End!\n";
    }
    return 0;
}
```

(E)Choice II

Description

在 DPY 的努力争取下，名额从 2 个变成了 3 个。好吧，Thor 又要重新计算了，囧。
他给 Thor 一段长度为 n 的数列和一个数 S，表示他对 n 个同学的看好程度 A[i]，要求 Thor 从里面选择三个人，使得这三个人的被看好程度之和大于 S。
现在，问 Thor 有多少的选择方式。

Input

多组数据。
每组数据有 2 行。
第一行有 2 个整数 n, s(1<=n<=1000, 0<=s<=200000)。
第二行有 n 个整数 A[i](0<=A[i]<=200000)表示每个人被看好程度。

Output

对于每组数据输出一行一个整数 Ans 表示 Thor 的选择方案数。

Sample Input

```
5 6
2 3 5 4 2
6 9
2 7 2 5 6 1
```

Sample Output

10
14

Hint

1000^3 是会超时的

解析：因为题目中要求找和大于 s 的三个数，所以应当优先考虑大的数字，也就是说需要从大到小排序。
易错点：如果按常规循环的话会超时，因此加一步判断，假如前两个数字之和已经不小于 s，那么不必再判断第三个数字。这样可以节省几乎一半的时间，因为两个大数字后面有很多小的数字。

参考代码：

```
#include<iostream>
#include<cstdio>
#include<cstdlib>

using namespace std;

int cmp(const void *a, const void *b)
{
    return *(int *)b - *(int *)a;
}

int main()
{
    int n, s, i, j, k;
    while (cin >> n >> s)
    {
        int A[1010] = { 0 };

        for (i = 0; i<n; i++)
            cin >> A[i];

        qsort(A, n, sizeof(int), cmp);
        //从大到小排序

        long long sum = 0;

        for (i = 0; i<n - 2; i++)
        {
            int bo = 0;
            for (j = i + 1; j<n - 1; j++)
```

```
            {
                if (A[i] + A[j] >= s)
                { //假如前两个数之和已经大于或等于s
                    //则不用再进行循环
                    sum = sum + n - 1 - j;
                    continue;
                }

                for (k = j + 1; k<n; k++)
                    if (A[i] + A[j] + A[k]>s)
                        sum++;
                //当三个数之和小于s时不必再判断更小的数字
                    else break;

            }
        }
        cout << sum << endl;
    }
    return 0;
}
```

(F)NeXT

Description

没办法，名额有限，没有被选中还有下一次嘛。
于是，DPY 给了 Thor 一个候选序列，为一个不带有重复字母且只有大写字母的字符串。
他让 Thor 变换一些序列中的顺序使得新得到的这个字符串是比原来的字符串大的字符串中最小的一个。
Thor 做不出来，可是幸好还有你。

Input

多组数据。
每组数据仅有 1 行。为一个不带有重复字母且只有大写字母的字符串。(保证字符串中出现的字符都是连续的，若长度为 3，那么最大的字符为 C)

Output

对于每组数据输出仅 1 行。
输出一个字符串，为题目中所描述的。如果找不到这样的字符串请输出"What?"。

Sample Input

A
ACB
CBA

Sample Output

What?
BAC
What?

Hint

关于字符串的比较规则即：两个字符串自左向右逐个字符相比（按 ASCII 值大小相比较），直到出现不同的字符或遇'\0'为止。如：
"A"<"B" "a">"A" "computer">"compare"

解析：
(1) 从最后一个字母开始查找，直到第一次找到这样一个字母 Ai，在其后面的字母有一个比该字母序号大;
(2) 找到在该字母后面、比该字母大且是最小的一个字母 Aj，将 Ai 与 Aj 交换位置;
(3) 对交换后的 Ai 之后的字母进行从小到大排序，即可得到结果

易错点：不能简单认为将某个较大的字母提前就可以了，它不能满足比前一个大且在两者之间无其他字符串。另外，注意什么时候输出"What?"

参考程序：

```
#include<iostream>
#include<cstring>
#include<algorithm>
using namespace std;

int cmp(const void *a, const void *b)
{
    return *((int *)a) - *((int *)b);
}

int main()
{
    char c[30];
    while (cin >> c)
    {
        int t = strlen(c), tt = t - 1, bo = 0, boo = 0, a, b, i, j;
        a = b = 29;
        c[29] = 160;
        //查找符合上述条件的字母
        for (i = t - 2; i >= 0; i--)
        {
            for (j = i + 1; j < t; j++)
            {
                if (c[j]>c[i]) bo++;
                if (bo)boo++;
                if (bo == 1 && boo == 1 || c[j]<c[b] && c[j]>c[a])
                {
                    a = i; b = j;
                }

            }
            if (bo)break;
        }

        if (!bo)
        {
            cout << "What?\n";
            continue;
        }
        else
        {
            char tem = c[a]; c[a] = c[b]; c[b] = tem;
            int A[30];
            //将char型数据转化为int型数据进行排序，
```

```
            //排序之后再转化为char型变量

            for (i = a + 1; i < t; i++)A[i - a - 1] = c[i];

            qsort(A, t - 1 - a, sizeof(int), cmp);

            for (i = a + 1; i < t; i++)c[i] = A[i - a - 1];
        }

        cout << c << endl;
    }
    return 0;
}
```

(G)胡乱走的和尚

题目描述：

从前有座山，山里有座庙，庙里有个不知姓名的老和尚和小和尚~~~o(∩_∩)o~~~
前两天，什么事情也没有发生呢~！0w0！~
这天，老和尚发现了一片错落有致的正方形区域上，如下图：

```
1   11 21 31 41 51 61 71 81 91
2   12 22 32 42 52 62 72 82 92
3   13 23 33 43 53 63 73 83 93
4   14 24 34 44 54 64 74 84 94
5   15 25 35 45 55 65 75 85 95
6   16 26 36 46 56 66 76 86 96
7   17 27 37 47 57 67 77 87 97
8   18 28 38 48 58 68 78 88 98
9   19 29 39 49 59 69 79 89 99
10 20 30 40 50 60 70 80 90 100
```

这居然还编号了！！！●▽●
方阵是从左上角的 1 按照图中顺序到右下角的 n^2
于是老和尚让小和尚蒙着眼站在 1 处，然后给他一系列指令
分别是向上(U)（或者左(L)，下(D)，右(R)）走整数 k 个格子
执行完这一系列的走位之后呢，看看小和尚知不知道自己站在几号格子上

小和尚需求大家程序的帮助啊！！！
（老和尚很阴险，有可能让小和尚走出方阵外，这种时候输出
"WanQuanGaoBuDong!"（不带引号）就好了

输入格式：

第一行为组数 T,T 为整数且 T<=10;
接下来 T 组，每组数据为 m+1 行，
第一行包含两个整数 n,m，表示方阵边长 n(1<=n<=10000)，指令数量
m(1<=n<=10000)；
下面 m 行，每行一个字母 x 和一个整数 k，字母 x 为上(U)、下(D)、右(R)、左(L)其中的一个，k(1<=k<=10000)为整数

输出格式：

对于每组测试数据，输出一行，为最后所在位置的数字，
如果在过程中出界了，输出"WanQuanGaoBuDong!"（不带引号）

sample in:

```
2
10 2
R 5
D 5
10 1
U 1
```

sample out:

```
56
WanQuanGaoBuDong!
```

解析：此题初看会认为需要用数组，其实并不需要；只要用两个变量保存小和尚当前位置，判断是否合法即可。
易错点：不能仅判断小和尚最终坐标，因为中间过程也会走出方阵。

参考程序：

```
#include<iostream>
using namespace std;
```

```
int main()
{
    int t, m, n, i, j, a; char c;
    cin >> t;
    while (t--)
    {
        cin >> n >> m;
        i = 1; j = 1; //初始时小和尚坐标
        int bo = 1;
        while (m--)
        {
            cin >> c >> a;
            if (c == 'U')i -= a;
            if (c == 'D')i += a;
            if (c == 'L')j -= a;
            if (c == 'R')j += a;

            if (i<1 || j<1 || i>n || j>n)
                bo = 0;
            //走出方阵
        }
        if (bo)
            cout << ((j - 1)*n + i) << endl;
        else
            cout << "WanQuanGaoBuDong!\n";
    }
    return 0;
}
```