D Chem Is A Fourth Try

——by 涨潮鹏 & Officially

题目描述

刚刚解开了 Lucifer•Tang 的第一层加密系统，Thor 与 Arthur 二人悲剧而又必然地发现——居然还有第二层！！！QAQ
幸运的是，第二层加密系统的解密模式与第一层大致相同，N 个序列的长度相同，且均不超过 1000。只不过这一次交换的模式，不一定只交换行，还可能交换列！
二人又看向了你……干巴爹——

输入

多组测试数据。
对于每组测试数据，第一行为三个数字 n，m，k(1≤n,m≤1000,1≤k≤100000)，用空格隔开。表示有 n 个序列，每个序列中有 m 个元素，总共有 k 次交换操作。
接下来 n 行，每行 m 个数，用空格隔开，为全部序列的情况。具体见样例。
再接下来 k 行，每行一个字符 c 与两个整数 i,j（保证 i,j 合法），用空格隔开。c 表示交换行或者列只会出现两种情况（行交换——R，列交换——C），i，j 表示对第 i、j 两行（或两列进行交换）。

输出

对于每组测试数据，输出最终的序列。格式见样例。

输入样例

3 3 3
1 2 3
4 5 6
7 8 9
C 1 2

C 2 3
R 1 2

输出样例

5 6 4
2 3 1
8 9 7

解题分析

题目要求对一个行列阵进行列交换或者行交换。可以这么想：

先简单点，就假设跟上次上机一样，只交换行。

那么，每一次交换后我们不知道某一行它到底跑哪去了。因此，便可以找一个变量来跟踪它在每次交换之后到哪儿去了。我想到的用来跟踪的变量（b[i]）是每次交换它所在的位置，即排在第 b[i]行。最后按照次序输出便可。（其实跟助教说的“没节操的序号”差不多的。）

借助第七次上机的“Chem Is A Third Try!!”的数据演示一下：

(1) 1

(2) 1 2

(3) 1 2 3

(4) 1 2 3 4

(5) 1 2 3 4 5

（1）、（5）交换后变成了：

（5）1 2 3 4 5

（2）1 2

（3）1 2 3

（4）1 2 3 4

（1）1

（2）、（5）交换后变成了：

（2）1 2

（5）1 2 3 4 5

（3）1 2 3

（4）1 2 3 4

（1）1

（3）、（4）交换后变成了：

（2）1 2

（5）1 2 3 4 5

（4）1 2 3 4

（3）1 2 3

（1）1

以上便得出了答案，其实前面的括号 2、5、4、3、1。便是对 b[5]={1,2,3,4,5}每次交换后里面所有元素的一个新

排列。

同理可得对列交换具有同样的规律和处理方法。

最后将两者综合起来就 AC 了。

参考代码：

```
#include<iostream>
#include<cstdio>
#include<cmath>
#include<algorithm>
#include<iomanip>
#include<cstring>
#include<string>
#define N 1010
int h[N],l[N];//h 代表行（hang）,l 代表列（lie）
int a[N][N];
using namespace std;
int main()
{
    int n,m,k;
    while(cin>>n>>m>>k)
    {   for(int i=1;i<=n;i++) h[i]=i;//给每一行赋初始值
        for(int i=1;i<=m;i++) l[i]=i;//给每一列赋初始值
```

```
        for(int i=1;i<=n;i++)
            for(int j=1;j<=m;j++)cin>>a[i][j];//读入整个阵
        for(int i=1;i<=k;i++)
        {
            char c;
            int p,q;
            cin>>c>>p>>q;
            if(c=='C')//交换列的位置。
            { int t=l[p];l[p]=l[q];l[q]=t; }
            else//交换行的位置。
            { int t=h[p];h[p]=h[q];h[q]=t; }
        }
        for(int i=1;i<=n;i++)
        {for(int j=1;j<m;j++)
          cout<<a[ h[i] ][ l[j] ]<<" ";
          cout<<a[ h[i] ][ l[m] ]<<endl;
       }
}


}
```

另一种思路：

其实另外一种思路与上一次上机的算法没有什么太大的区别，仍然是完全使用指针进行处理，考虑行变换与列变换的顺序实际上是不影响结果的，于是将所有的操作存下来，首先进行完毕全部的行变换，然后将所有的数字转向另外一个指针数组中，在其中进行列变换，即可得到最后的正确结果。

参考代码：

```
#include<iostream>
#include<cstdio>
using namespace std;
int n,m,k,*p1[1001],*p2[1001],a,b,*tmp;
int colS[10001][2],rowS[10001][2],numC,numR;
char c;
int main()
{
    while(scanf("%d%d%d",&n,&m,&k)!=EOF)
    {
        numC=numR=0;
        for(int i=0;i<n;i++)
           p1[i]=new int[m];
        for(int i=0;i<m;i++)
           p2[i]=new int[n];
        for(int i=0;i<n;i++)
           for(int j=0;j<m;j++)
              scanf("%d",&p1[i][j]);
        for(int i=0;i<k;i++)
        {
            scanf(" %c %d%d",&c,&a,&b);
            if(c=='R')    {rowS[numR][0]=a-1;rowS[numR++][1]=b-1;}
            else          {colS[numC][0]=a-1;colS[numC++][1]=b-1;}
        }
        for(int i=0;i<numR;i++)
        {
            tmp=p1[rowS[i][0]];
            p1[rowS[i][0]]=p1[rowS[i][1]];
            p1[rowS[i][1]]=tmp;
        }
        for(int i=0;i<n;i++)
           for(int j=0;j<m;j++)
```

```
            p2[j][i]=p1[i][j];
        for(int i=0;i<numC;i++)
        {
            tmp=p2[colS[i][0]];
            p2[colS[i][0]]=p2[colS[i][1]];
            p2[colS[i][1]]=tmp;
        }
        for(int i=0;i<n;i++)
        {
           for(int j=0;j<m;j++)
              printf("%d ",p2[j][i]);
           printf("\n");
        }
        for(int i=0;i<n;i++)
           delete[] p1[i];
        for(int i=0;i<m;i++)
           delete[] p2[i];
    }
}
```