**猪的安家**

Andy和Mary养了很多猪。他们想要给猪安家。但是Andy没有足够的猪圈，很多猪只能够在一个猪圈安家。举个例子，假如有16头猪，Andy建了3个猪圈，为了保证公平，剩下1头猪就没有地方安家了。Mary生气了，骂Andy没有脑子，并让他重新建立猪圈。这回Andy建造了5个猪圈，但是仍然有1头猪没有地方去，然后Andy又建造了7个猪圈，但是还有2头没有地方去。Andy都快疯了。你对这个事情感兴趣起来，你想通过Andy建造猪圈的过程，知道Andy家至少养了多少头猪。  
  
输入  
  
输入包含多组测试数据。每组数据第一行包含一个整数n (n <= 10) – Andy建立猪圈的次数，解下来n行，每行两个整数ai, bi( bi <= ai <= 1000), 表示Andy建立了ai个猪圈，有bi头猪没有去处。你可以假定(ai, aj) = 1.  
  
输出  
  
输出包含一个正整数，即为Andy家至少养猪的数目。  
  
样例输入  
  
3  
3 1  
5 1  
7 2  
  
样例输出  
  
16

答案:

// 猪的安家.cpp : Defines the entry point for the console application.  
//  
  
#include "stdafx.h"  
#include "iostream.h"   
  
  
void main()  
{  
    int n;  
    int s[10][2];  
    bool r[10];  
    char ch;  
  
    cout<<"请输入次数："<<endl;  
    cin>>n;  
      
    for (int i=0;i<n;i++)  
    {  
        cout<<"请输入第"<<i+1<<"次的猪圈个数和剩下的猪：（用--分开）"<<endl;  
        cin>>s[i][0]>>ch>>ch>>s[i][1];  
    }  
  
    for (i=0;i<10;i++)  
        r[i]=true;  
  
  
    for (int sum=1;;sum++)  
    {  
        for (i=0;i<n;i++)  
            r[i]=(sum%s[i][0]==s[i][1]);  
        for (i=0;i<n;i++)  
        {  
            if (r[i]==0)  
                break;  
        }  
        if (i==n)  
        break;  
    }  
    cout<<"猪至少有"<<sum<<"只。"<<endl;  
      
  
}

**蛇行矩阵**

Problem  
蛇形矩阵是由1开始的自然数依次排列成的一个矩阵上三角形。   
  
Input  
本题有多组数据，每组数据由一个正整数N组成。（N不大于100）   
  
Output  
对于每一组数据，输出一个N行的蛇形矩阵。两组输出之间不要额外的空行。   
  
矩阵三角中同一行的数字用一个空格分开。行尾不要多余的空格。   
  
Sample Input  
5  
  
Sample Output  
1 3 6 10 15  
2 5 9 14  
4 8 13  
7 12  
11

答案:

//SNAKE 矩阵  
////////////////////////////////////////////////////////////////////////////////  
  
#define M 100  
  
int main()  
{  
    int a[M][M];  
    int n;  
    printf("请输入N的值：\n");  
    scanf("%d",&n);  
    if(n <= 0)  
        return 0;  
    printf("形成的蛇形矩阵为：\n");  
    a[0][0] = 1;//因为每次第一个数必是1   
    printf("%d ",a[0][0]);  
    for(int i=0;i<n-1;i++)  
    {  
        a[i+1][0] = a[i][0] + i + 1;//提前算出下一行的第一个数据  
        for(int j=0;j<n-i-1;j++)  
        {  
            a[i][j+1] = a[i][j] + j + i + 2;//算出每一行的每个数据  
            printf("%d ",a[i][j+1]);//打印每一行的数据  
        }  
        printf("\n%d ",a[i+1][0]);//打印下一行的第一个数据  
    }  
    printf("\n");  
    return 0;  
}

**平方数**

给出包含M个数字的列表，和列表中所有数字的所有质因数。求出最长的子列表，使得子列表中所有数字的乘积是一个完全平方数。   
  
输入  
  
输入文件包含多组测试数据。第一行包含两个整数N , M ( 1 <= N <= 30 , 1 <= M <= 30000 ). N 是质因数的个数。接下来一行有N个整数，给出所有的质因数。然后一行包含M个整数，给出列表。   
  
输入文件结束于N = M = 0.   
  
输出  
  
对于每组数据，输出最长子列表的两个位置坐标l r。l是该子列表在列表中的起始位置，r是结束位置。如果多种情况都满足子列表长度最大，输出l最小的一个。如果不存在这样的子列表输出“None”。   
  
样例输入  
  
3 4  
2 3 5  
4 9 25 6  
3 4  
2 3 5  
6 6 3 3  
0 0  
  
样例输出  
  
1 3  
1 4

答案:

#include <iostream>  
  
using namespace std;  
  
  
#define MAX\_N 30  
#define MAX\_M 30000   
  
void squareNumber(int \*A,int \*B,int N,int M);  
  
void main()  
{  
  
    int A[MAX\_N];  
    int B[MAX\_M];  
    int N,M;  
    cin>>N>>M;  
    int i;  
  
    while (N!=0||M!=0)  
    {  
        for (i=0;i<N;i++)  
        {  
            cin>>A[i];  
        }  
        for (i=0;i<M;i++)  
        {  
            cin>>B[i];  
        }  
        squareNumber(A,B,N,M);  
  
        cin>>N>>M;  
          
          
    }  
  
}  
  
  
void squareNumber(int \*A,int \*B,int N,int M)  
{  
    int i,j,k,t;  
    bool flag;  
  
    int num[MAX\_N]={0};  
    int temp=1;  
  
    int len\_max=0;  
    int r=0;  
    int f=0;  
  
  
    for (i=0;i<M;i++)  
    {  
          
        for (j=i+1+len\_max;j<=M;j++)  
        {  
              
            for (k=0;k<N;k++)  
            {  
                num[k]=0;  
            }  
            for (k=i;k<j;k++)  
            {  
                temp=B[k];  
                while (temp!=1)  
                {  
                    flag=false;  
                    for (t=0;t<N;t++)  
                    {  
                        if (temp%A[t]==0)  
                        {  
                            temp/=A[t];  
                            num[t]++;  
                            flag=true;  
                        }  
                    }  
                    if (!flag)  
                    {  
                        cout<<"&Ecirc;&yacute;&frac34;&Yacute;&Oacute;&ETH;&Icirc;ó"<<endl;  
                        return;  
                    }  
                }  
            }  
            flag=true;  
  
            for (k=0;k<N;k++)  
            {  
  
                if (num[k]%2==1)  
                {  
                    flag=false;  
                    break;  
                }  
            }  
            if (flag)  
            {  
                len\_max=j-i;  
                f=i+1;  
                r=j;  
  
                  
            }  
  
        }  
    }  
    cout<<f<<" "<<r<<endl;  
  
}

**死亡迷宫**

背景  
  
很久以前，迷宫里住着一个恶魔。一天，我们伟大的英雄Andy无意中踏入了这个迷宫。不幸的是，他被困在这个迷宫当中了。恶魔在迷宫中召唤出了许多怪物，想要阻止Andy逃脱。在迷宫中，Andy遇到一个一位巫师。他给了Andy迷宫的地图，并告诉他迷宫的入口很快会关闭。Andy必须以非常快的速度到达入口，并且有足够的力气推开挡在入口的岩石。于是，Andy带着地图一路向着出口走去……   
  
问题  
  
给出Andy和各怪物的能量, 攻击力, 防御力，和迷宫的地图，请你计算一下 能量/耗时 的最大值。   
  
当Andy走到有怪物的地方时，Andy会先进行攻击，然后怪物攻击，然后Andy……当一方的能量小于等于0时攻击停止，并且小于等于0的一方死亡。攻击时，每次对方损耗的能量为己方的攻击力减去对方的防御力。   
  
当Andy走到标有‘A’,‘B’,‘C’的地方时，Andy的相应属性会得到增加。   
  
  
对应关系如下：  
  
[A] 能量 + P  
[B] 攻击力 + Q  
[C] 防御力 + R  
  
如果耗时超过100，那么门将永远也打不开了，我们的Andy也就永远的困在了这个暗无天日的迷宫之中……   
  
输入  
  
标准输入包含多组数据。  
  
每组数据的第一行有六个整数W (1 <= W <= 20), H (1 <= H <= 20), P (1 <= P <= 10), Q (1<= Q <= 10), R (1 <= R <= 10), M (0 <= M <= 5). 迷宫是由一个W\*H的矩形区域构成。M表示怪物的数量。Andy每个单位时间可以移动到相邻的4个格中，当然，必须得保证目标格在矩形区域中。默认的起始时间是0。与怪物战斗不会花费额外的时间。  
  
其后H行每行严格包含W个字符。用如下的各字符表示这个迷宫的地图：  
  
[#]表示一堵墙（Andy是不会穿墙术的）  
[.] Marks an empty space, into which you can move.表示一块空地。  
[S]表示Andy的初始位置。  
[E]表示迷宫的入口。  
[0]表示各怪物。  
[A]表示属性增加地点。（使用次数仅限于一次）  
  
其后一行有三个整数，表示Andy的能量，攻击力，和防御力。  
  
其后M行，每行有四个整数，表示怪物的编号，和这个怪物的各属性。  
  
输出  
  
对于每组输入数据，输出 能量/耗时 的最大值，并保留4位小数。如果Andy不能到达出口，输出“impossible”。数据之间无空行。  
  
样例输入  
  
6 17 7 5 4 3  
#################  
##E......#......#  
#A#....#.0.##.#B#  
#1###########2###  
#.S............C#  
#################  
100 59 10  
0 23 48 0  
1 65 41 0  
2 20 27 0  
  
样例输出  
  
3.7037

答案:

#include <iostream>  
#include <fstream>  
#include <queue>  
using namespace std;  
#define MAX\_MAP\_SIZE 20  
#define MAX\_MON\_SIZE 5  
#define Time 100  
int W,H,P,Q,R,M;  
char map[MAX\_MAP\_SIZE][MAX\_MAP\_SIZE];  
   
  
typedef struct Andy   
{  
 int p;  
 int q;  
 int r;  
} \*pAndy;  
typedef struct Monster  
{  
 int id;  
 int p;  
 int q;  
 int r;  
}\*pMonster;  
typedef struct Node  
{  
 int x;  
 int y;  
}\* pNode;  
  
Andy andy;  
Monster mons[MAX\_MON\_SIZE];  
char keywords[4+MAX\_MON\_SIZE];  
int shortestPath(pNode path,int len\_path,char des,char \* revs,int len\_rev,int time);  
float escape(char \*revs,int len\_revs,int time,Andy andy);  
bool getPosition(char ch,pNode p);  
void main()  
{  
 ifstream in("data.in");  
 in>>W>>H>>P>>Q>>R>>M;  
 int i;  
 for (i=0;i<W;i++)  
 {  
  in>>map[i];  
  //cout<<map[i]<<endl;  
    
 }  
 in>>andy.p>>andy.q>>andy.r;  
  
 for(i=0;i<MAX\_MON\_SIZE;i++)  
 {  
  in>>mons[i].id;  
  in>>mons[i].p;  
  in>>mons[i].q;  
  in>>mons[i].r;  
 }  
 in.close();  
 keywords[0]='E';  
 keywords[1]='A';  
 keywords[2]='B';  
 keywords[3]='C';  
 for (i=0;i<M;i++)  
 {  
  keywords[4+i]='0'+i;  
 }  
 /\*Node path[MAX\_MAP\_SIZE\*MAX\_MAP\_SIZE];  
 path[0].x=4;  
 path[0].y=2;  
 int len\_path=1;  
 char des='E';  
 char revs[10]=".012ABCS";  
 int len\_rev=8;  
 int time=100;  
 cout<<shortestPath(path,len\_path,des,revs,len\_rev,time)<<endl;\*/  
  
 char revs[10]=".S";  
 int len\_revs=2;  
 int time=100;  
  
 cout<<escape(revs,len\_revs,time,andy)<<endl;  
  
 /\*char revs[10]=".S20E";  
 int len\_revs=5;  
 Node path[MAX\_MAP\_SIZE\*MAX\_MAP\_SIZE];  
 int len\_path=1;  
   
 int time=100;  
    
  for(int j=1;j<4;j++)  
  {  
      getPosition(revs[j],path);  
      cout<<shortestPath(path,len\_path,revs[j+1],revs,j+1,time)<<endl;  
  
  }\*/  
   
   
}  
int shortestPath(pNode path,int len\_path,char des,char \* revs,int len\_rev,int time)  
{  
 if (len\_path==0)  
 {  
  return -1;  
 }  
 pNode p=path+len\_path-1;  //&micro;±&Ccedil;°&frac12;&Uacute;&micro;&atilde;  
  
 if (map[p->x][p->y]==des) //&Otilde;&Ograve;&micro;&frac12;&Auml;&iquest;±ê   
 {  
  return len\_path-1;     
 }  
 if (time<1)  //&Atilde;&raquo;&Oacute;&ETH;&Ecirc;±&frac14;&auml;&frac14;&Igrave;&ETH;&oslash;&Ntilde;°&Otilde;&Ograve;  
 {  
  return -1;  
 }  
 int i,j,k;  
 for (i=0;i<len\_rev;i++)  
 {  
  if (map[p->x][p->y]==revs[i])  
  {  
   Node nodes[4]={{p->x,p->y-1},{p->x,p->y+1},{p->x-1,p->y},{p->x+1,p->y}};  
   int min=W\*H;  
   int temp;  
   int bid=-1;  
   bool flag;  
     
   for (j=0;j<4;j++)  
   {  
    flag=true;  
    for (k=0;k<len\_path-1;k++)  
    {  
     if(path[k].x==nodes[j].x&&path[k].y==nodes[j].y)  
     {  
      flag=false;   
      break;  
     }  
    }  
    if (flag)  
    {  
     path[len\_path]=nodes[j];  
     temp=shortestPath(path,len\_path+1,des,revs,len\_rev,time-1);  
     if (temp>0&&temp<min)  
     {  
      min=temp;  
      bid=i;  
     }  
    }  
   }  
   return bid==-1? -1:min;    
  }  
 }  
 return -1;  
}  
  
bool getPosition(char ch,pNode p)  
{  
    for (int i=0;i<W;i++)  
    {  
        for (int j=0;j<H;j++)  
        {  
            if (map[i][j]==ch)  
            {  
                p->x=i;  
                p->y=j;  
                return true;  
            }  
        }  
    }  
    return false;  
}  
  
float escape(char \*revs,int len\_revs,int time,Andy andy)  
{  
  
    int i,j;  
    int tempTime;  
    float max=-1;  
    float  temp;  
    Node path[MAX\_MAP\_SIZE\*MAX\_MAP\_SIZE];  
    int len\_path=1;  
    Andy newAndy;  
    bool flag;  
    for (i=0;i<4+M;i++)  
    {  
        flag=false;  
        for (j=1;j<len\_revs;j++)  
        {  
            if (revs[j]==keywords[i])  
            {  
                flag=true;  
                break;  
                  
            }  
        }  
        if (flag)  
        {  
            continue;  
        }  
          
        getPosition(revs[len\_revs-1],path);  
        tempTime=shortestPath(path,len\_path,keywords[i],revs,len\_revs,time);  
  
        if (tempTime>0)  
        {  
            if (keywords[i]=='E')   //&Otilde;&Ograve;&micro;&frac12;&sup3;&ouml;&iquest;&Uacute;  
            {  
                temp=(float)andy.p/(Time-time+tempTime);  
  
                /\*cout<<temp<<" "<<andy.p<<" "<<time<<" ";  
                for (int m=0;m<len\_revs;m++)  
                {  
                    cout<<revs[m];  
                }  
                cout<<endl<<endl;\*/  
                //cout<<temp<<endl;  
            }  
            else  
            {  
                  
                if (keywords[i]=='A')  
                {  
                    newAndy.p=andy.p+P;  
                    newAndy.q=andy.q;  
                    newAndy.r=andy.r;  
  
                }  
                else if (keywords[i]=='B')  
                {  
                    newAndy.p=andy.p;  
                    newAndy.q=andy.q+Q;  
                    newAndy.r=andy.r;  
                }  
                else if (keywords[i]=='C')  
                {  
                    newAndy.p=andy.p;  
                    newAndy.q=andy.q;  
                    newAndy.r=andy.r+R;  
                }  
                else    //&Ocirc;&acirc;&Oacute;&ouml;&sup1;&Ouml;&Ecirc;&THORN;  
                {  
                    pMonster pMon=mons+keywords[i]-'0';  
  
                    if (andy.q<pMon->r)  
                    {  
                        newAndy.p=-1;  //&cedil;ù±&frac34;&sup1;&yacute;&sup2;&raquo;&Egrave;&yen;&pound;&not;&sup1;&Ouml;&Ecirc;&THORN;&micro;&Auml;·&Agrave;&Oacute;ù&Igrave;&laquo;&cedil;&szlig;  
                    }  
                    else if (pMon->q<andy.r)   //andy&micro;&Auml;·&Agrave;&Oacute;ù&ordm;&Uuml;&cedil;&szlig;&pound;&not;&para;&oslash;&sup2;&raquo;&raquo;á&Euml;&eth;&Ecirc;§&Auml;&Uuml;&Aacute;&iquest;  
                    {  
                        newAndy.p=andy.p;      
  
                    }  
                    else  
                    {  
                        newAndy.p=andy.p-(pMon->q-andy.r)\*(pMon->p/(andy.q-pMon->r));  
                    }  
  
                  
                    newAndy.q=andy.q;  
                    newAndy.r=andy.r;  
                }  
                revs[len\_revs]=keywords[i];  
                temp=escape(revs,len\_revs+1,time-tempTime,newAndy);  
            }  
            max=max>temp? max:temp;          
        }  
  
    }  
    return max;  
  
}