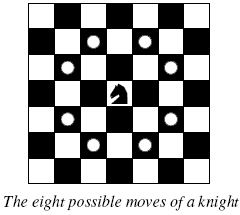
**A Knight's Journey**

|  |  |  |
| --- | --- | --- |
| **Time Limit:** 1000MS |  | **Memory Limit:** 65536K |
| **Total Submissions:** 14095 |  | **Accepted:** 4692 |

**Description**

**Background**   
The knight is getting bored of seeing the same black and white squares again and again and has decided to make a journey   
around the world. Whenever a knight moves, it is two squares in one direction and one square perpendicular to this. The world of a knight is the chessboard he is living on. Our knight lives on a chessboard that has a smaller area than a regular 8 \* 8 board, but it is still rectangular. Can you help this adventurous knight to make travel plans?   
  
**Problem**   
Find a path such that the knight visits every square once. The knight can start and end on any square of the board.

**Input**

The input begins with a positive integer n in the first line. The following lines contain n test cases. Each test case consists of a single line with two positive integers p and q, such that 1 <= p \* q <= 26. This represents a p \* q chessboard, where p describes how many different square numbers 1, . . . , p exist, q describes how many different square letters exist. These are the first q letters of the Latin alphabet: A, . . .

**Output**

The output for every scenario begins with a line containing "Scenario #i:", where i is the number of the scenario starting at 1. Then print a single line containing the lexicographically first path that visits all squares of the chessboard with knight moves followed by an empty line. The path should be given on a single line by concatenating the names of the visited squares. Each square name consists of a capital letter followed by a number.   
If no such path exist, you should output impossible on a single line.

**Sample Input**

3

1 1

2 3

4 3

**Sample Output**

Scenario #1:

A1

Scenario #2:

impossible

Scenario #3:

A1B3C1A2B4C2A3B1C3A4B2C4

**Source**

[TUD Programming Contest 2005](http://poj.org/searchproblem?field=source&key=TUD+Programming+Contest+2005), Darmstadt, Germany

//深搜练习题。

#include<iostream>

#include<vector>

using namespace std;

class Position

{

public:

int x,y;

};

vector<Position> path;

int dx[8] = {-1, 1, -2, 2, -2, 2, -1, 1};

int dy[8] = {-2, -2, -1, -1, 1, 1, 2, 2};

int len,n,m;

bool flag=false;

int a[30][30];

void dfs(Position now)

{

a[now.x][now.y]=1;

Position next;

for(int i=0;i<8;i++)

{

next.x=now.x+dx[i];

next.y=now.y+dy[i];

if(next.x>=1 && next.x<=m && next.y>=1 && next.y<=n && a[next.x][next.y]==0)

{

path.push\_back(next);

if(path.size()==len)

{flag=true;break;}

dfs(next);

if(flag)

return ;

a[next.x][next.y]=0;

path.pop\_back();

}

}

}

int main()

{

int t,i,j,k;

cin>>t;

for(k=1;k<=t;k++)

{

cin>>m>>n;

for(i=1;i<=m;i++)

for(j=1;j<=n;j++)

a[i][j]=0;

flag=false;

len=m\*n;

path.clear();

Position begin;

begin.x=begin.y=1;

path.push\_back(begin);

if(n==1 && m==1)

flag=1;

dfs(begin);

cout<<"Scenario #"<<k<<":\n";

if(flag)

{

for(i=0;i<len;i++)

cout<<char(path[i].y-1+'A')<<path[i].x;

cout<<endl;

}

else

cout<<"impossible\n";

if(k!=t)

cout<<endl;

}

return 0;

}