**RAT IN A MAZE**

You are given an N\*M grid. Each cell (i,j) in the grid is either blocked, or empty. The rat can move from position (i,j), down or right on the grid.  
Initially rat is on the position (1,1). It wants to reach position (N,M). Find the rightmost path through which, rat can reach this position. A path is rightmost, if the rat is at position (i,j), it will always move to (i,j+1), if there exists a path from (i,j+1) to (N,M).

**Input Format:**

First line contains 2 integers, N and M, denoting the rows and columns in the grid. Next N line contains. M characters each. An 'X' in position (i,j) denotes that the cell is blocked and ans 'O' denotes that the cell is empty.

**Constraints:**

1<=N,M<=1000 GRID(i,j)='X' or 'O'

**Output Format**

If a solution exists: Print N lines, containing M integers each. A 1 at a position (i,j) denotes that the (i,j)th cell is covered in the path and a 0 denotes that the cell is not covered in the path.  
If solution doesn't exist, just print "-1".

**Sample Input**

5 4

OXOO

OOOX

OOXO

XOOO

XXOO

**Sample Output**

1 0 0 0

1 1 0 0

0 1 0 0

0 1 1 1

0 0 0 1

Program-

#include<iostream>

using namespace std;

bool ratInMaze(char maze[1001][1001],int sol[1001][1001],int i,int j,int m,int n){

// base case

if(i==m && j==n){

sol[m][n]=1;

// printing the sol array

for(int i=0;i<=m;i++){

for(int j=0;j<=n;j++){

cout<<sol[i][j]<<" ";

}

cout<<endl;

}

cout<<endl;

return true;

}

// rec case

/// rat out of the grid

if(i>m || j>n){

return false;

}

if(maze[i][j]=='X'){

return false;

}

sol[i][j]=1;

bool forwardSuccess=ratInMaze(maze,sol,i,j+1,m,n);

bool down;

if(!forwardSuccess)

{

down=ratInMaze(maze,sol,i+1,j,m,n);

}

sol[i][j]=0; // backtracking

if(forwardSuccess||down){

return true;

}

return false;

}

int main() {

int m,n;

cin>>m>>n;

char maze[1001][1001];

for(int i=0;i<m;i++){

for(int j=0;j<n;j++){

cin>>maze[i][j];

}

}

int sol[1001][1001]={0};

bool ans=ratInMaze(maze,sol,0,0,m-1,n-1);

if(ans==false){

cout<<"-1";

}

}