Description | Hints | Submissions | Discussions | Notes

KWALK

[illegible] Difficulty | 4 sec Time Limit | 256000KB Memory | 100 Score

80/80 XP | 30/30

Description

You are given an $N \times N$ chessboard and a knight with starting position *(Sx, Sy)*. You are given a final position *(Fx, Fy)*. You have to find the minimum number of moves required to reach the final position.

Complete the function

```
int KnightWalk(int N, int Sx, int Sy, int Fx, int Fy)
```

Copy

Input Format

The first line contains a single integer *T* - the number of test cases.

The first line of each test case contains five integers *N Sx Sy Fx Fy* - the size of the board, initial position and final position.

Output Format

For every test case print the minimum number of moves required. If it is not possible print -1.

Constraints

1 ≤ T ≤ 20

1 ≤ N ≤ 1000

1 ≤ Sx, Sy, Fx, Fy ≤ N

Sample Input 1 Copy

```
3
6 4 5 1 1
```

C++14 | 00:00:00 | 12 px

```
#include <bits/stdc++.h>
using namespace std;
#define endl "\n"
using lli = long long int;
using pp = pair<int, int>;
vector<vector<int>> dis;
vector<vector<int>> vis;
int N, Sx, Sy, Fx, Fy;

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


bool check(int x, int y)
{
    if (x >= 1 && y >= 1 && x <= N && y <= N)
    {
        return true;
    }
    return false;
}

vector<pp> neighbour(pp p)
{
    vector<pp> ans;
    for (int i = 0; i < 8; i++)
    {
        int x = p.first + dx[i];
        int y = p.second + dy[i];
        if (check(x, y))
        {
            ans.push_back(make_pair(x, y));
        }
    }
    return ans;
}
```

Sample Tests | Manual Tests

Console | Run on Sample