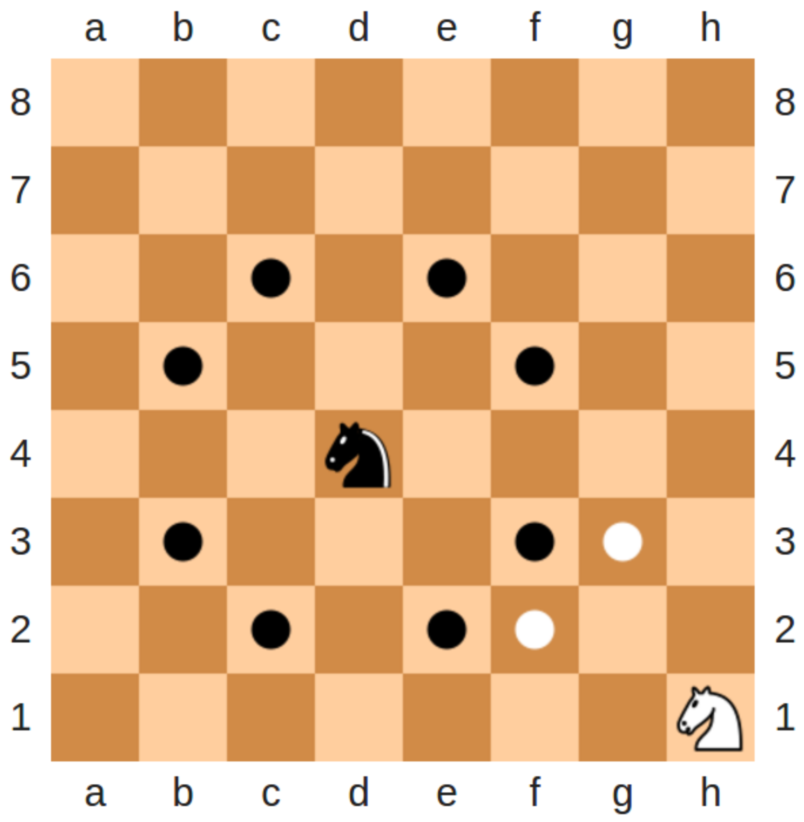
# Project: Traveling Knight

## Problem Description

Given a chess board, your job is to write a program that takes two squares `x` and `y` as input and then determines the number of knight moves on a **shortest** route from `x` to `y`.



## Input Specification

Your program should read from an input file, which will contain one or more test cases. Each test case consists of one line containing two squares separated by one space. A square is a string consisting of a letter (a-h) representing the column and a digit (1-8) representing the row on the chessboard.

### Sample Input

```
e2 e4
a1 b2
b2 c3
a1 h8
a1 h7
h8 a1
b1 c3
f6 f6
```

## Output Specification

For each test case, print one line saying `To get from xx to yy takes ?? knight moves.` in console.

### Sample Output

Below is the correct output for the previous sample input.

```
To get from e2 to e4 takes 2 knight moves.
To get from a1 to b2 takes 4 knight moves.
To get from b2 to c3 takes 2 knight moves.
To get from a1 to h8 takes 6 knight moves.
To get from a1 to h7 takes 5 knight moves.
To get from h8 to a1 takes 6 knight moves.
To get from b1 to c3 takes 1 knight moves.
To get from f6 to f6 takes 0 knight moves.
```

## Hint

- According to the problem description, the grid is 8 by 8, therefore there are 64 squares. We can then model the chessboard as a graph with 64 vertices; and an edge between two vertices exists if and only if we can make a knight move from one vertex to another.
- When moving to the next position, a knight can have **at most eight** possible directions, as shown in the above figure for the black knight, without getting out of the chess board boundary.
- The problem becomes a graph traversal problem!
- DFS or BFS? Which one will work?

Good luck!