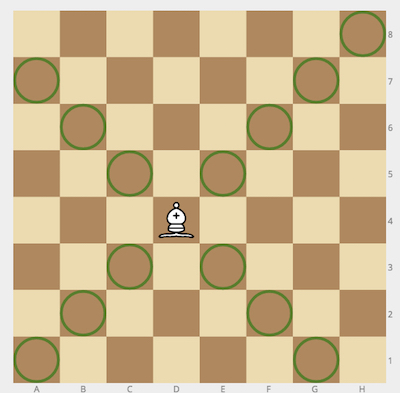
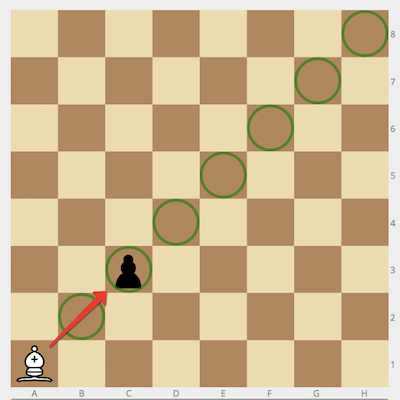
Given the positions of a white bishop and a black pawn on the standard chess board, determine whether the bishop can capture the pawn in one move.

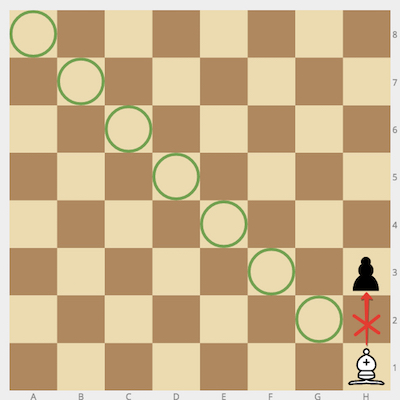
The bishop has no restrictions in distance for each move, but is limited to diagonal movement. Check out the example below to see how it can move:  


Example

* For bishop = "a1" and pawn = "c3", the output should be  
  bishopAndPawn(bishop, pawn) = true.



* For bishop = "h1" and pawn = "h3", the output should be  
  bishopAndPawn(bishop, pawn) = false.



Input/Output

* **[execution time limit] 0.5 seconds (cpp)**
* **[input] string bishop**

Coordinates of the white bishop in the [chess notation](keyword://chess-notation" \t "_blank).

* **[input] string pawn**

Coordinates of the black pawn in the same notation.

* **[output] boolean**
  + true if the bishop can capture the pawn, false otherwise.

**[C++] Syntax Tips**

// Prints help message to the console

// Returns a string

std::string **helloWorld**(std::string name) {

std::cout << "This prints to the console when you Run Tests" << std::endl;

**return** "Hello, " + name;

}

<https://app.codesignal.com/arcade/code-arcade/chess-tavern/6M57rMTFB9MeDeSWo/description?solutionId=zcqzhATt78Y95dpK7>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Program

{

static bool bishopAndPawn(string bishop, string pawn)

{

char letraBishop = bishop[0];

char numBishop = bishop[1];

char letraPawn = pawn[0];

char numPawn = pawn[1];

return Math.Abs((int)letraBishop - (int)letraPawn) ==

Math.Abs((int)(numBishop) - (int)numPawn);

}

static void Main(string[] args)

{

Console.ReadLine();

}

}

}