IF100 – Fall 2022-2023
Take-Home Exam 2
Deadline: November 18th 2022, Friday, 23:59

Introduction

The aim of this take-home exam is to practice decision making (conditional if statements), strings and methods. The use of if statements is due to the nature of the problem; that is, you cannot finish this homework without using decision making. Specifically for this assignment, it will allow you to successfully complete the essential tasks while also highlighting errors by providing the appropriate warnings to the user at the right time.

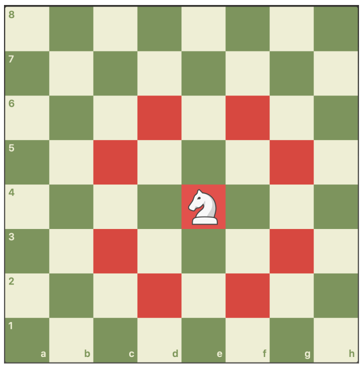
Description

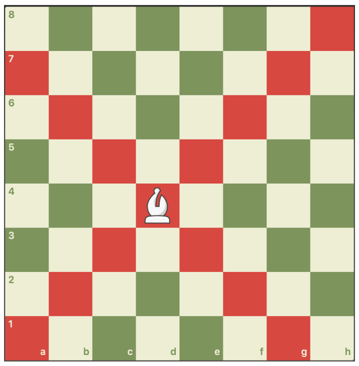
One of the biggest debates in chess has always been whether the knight or the bishop is worth more. Despite the fact that many great chess masters provide diverse solutions to this question, the answer still changes based on the positions of the chess pieces on the chessboard. Let's make a contribution to this debate, which cannot be resolved despite dozens of books being written about it.

In this task, we should create a Python program that determines whether a bishop and a knight on a chessboard are placed in such a way that they can attack each other considering their locations and relative positions against each other on the chessboard.

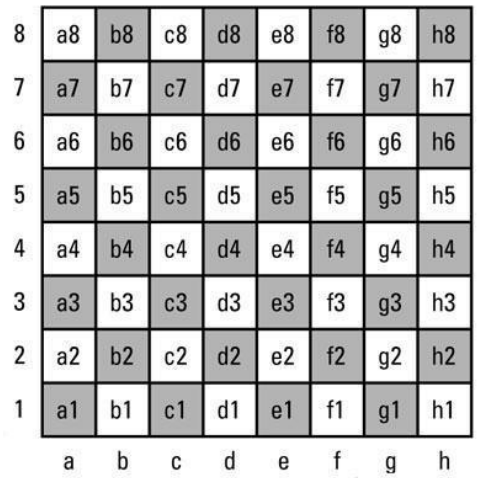
In a chess game, the bishop moves only on diagonals, while the knight moves one square vertically and two squares horizontally, or two squares vertically and one square horizontally. This movement is known as an "L-shape" because it resembles the capital letter "L."

Below images show an example of the squares that a knight and a bishop can move to, considering their current positions.





The chess board is made up of 64 contrasting colored squares, and it is divided into eight horizontal ranks (from 1 to 8) and eight vertical files (from a to h), allowing each of the 64 squares to be identified using those of the letters and digits mentioned.



8 a8 b8 c8 d8 e8 f8 g8 h8
7 a7 b7 c7 d7 e7 f7 g7 h7
6 a6 b6 c6 d6 e6 f6 g6 h6
5 a5 b5 c5 d5 e5 f5 g5 h5
4 a4 b4 c4 d4 e4 f4 g4 h4
3 a3 b3 c3 d3 e3 f3 g3 h3
2 a2 b2 c2 d2 e2 f2 g2 h2
1 a1 b1 c1 d1 e1 f1 g1 h1
a b c d e f g h

To determine whether or not the chess pieces will attack each other, we need to get the chess pieces' positions from the user as a set of inputs. And the program will then check whether those inputs are valid or not, based on the conditions provided in the next section. Whenever the user enters an invalid input, your program should display an appropriate error message and immediately stop its execution without further asking for inputs and making any decisions. On the other hand, if all the inputs are valid, then the program will decide and display the result of whether the knight can attack the bishop or the bishop can attack the knight or none can attack each other.

Input, Input Check and Output

The inputs of the program and their order are explained below. It is extremely important to follow this order with the same characters since we automatically process your programs. ***Thus, your work will be graded as 0 unless the order is entirely correct***. Please see the "Sample Runs" section for some examples.

The prompts of the input statements to be used has to be exactly the same as the prompts of the "Sample Runs".

Here is detailed information on inputs and input checks:

- Horizontal position of the knight (a,b,c,d,e,f,g,h)
    - If not a -single- letter, then your program should display:
      Horizontal input for knight is not a letter
    - If it is a single letter, but not one of the letters that could be on the chessboard (a,b,c,d,e,f,g,h), then your program should display:
      Horizontal input for knight is not a proper letter
    - The letter should be considered in a case-insensitive way, so A,B,C,D,E,F,G,H should also be considered as valid inputs.

- Vertical position of the knight (1,2,3,4,5,6,7,8)
    - If not a non-negative integer, then your program should display:
      Vertical input for knight is not a number
    - If it is a non-negative integer, but not one of the numbers that could be on the chessboard (1,2,3,4,5,6,7,8), then your program should display:
      Vertical input for knight is not a proper number

- Horizontal position of the bishop (a,b,c,d,e,f,g,h)
    - If not a -single- letter, then your program should display:
      Horizontal input for bishop is not a letter
    - If it is a single letter, but not one of the letters that could be on the chessboard (a,b,c,d,e,f,g,h), then your program should display:
      Horizontal input for bishop is not a proper letter
    - The letter should be considered in a case-insensitive way, so A,B,C,D,E,F,G,H are also valid inputs.

- Vertical position of the bishop (1,2,3,4,5,6,7,8)
    - If not a non-negative integer, then your program should display:
      Vertical input for bishop is not a number
    - If it is a non-negative integer, but not one of the numbers that could be on the chessboard (1,2,3,4,5,6,7,8), then your program should display:
      Vertical input for bishop is not a proper number

If both the bishop and the knight are inputted in the same frame, then your program should display:

They can't be in the same square

If the knight can attack the bishop, then your program should display:

Knight can attack bishop

If the bishop can attack the knight, then your program should display:

Bishop can attack knight

If neither of the knight or the bishop can attack each other, then your program should display:

Neither of them can attack each other

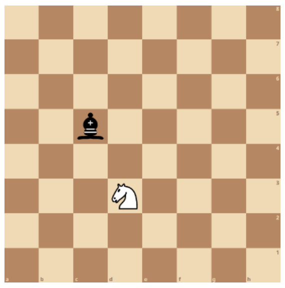
You may check the "Sample Runs" section given below for some examples.

Sample Runs

Below, we provide some sample runs of the program that you will develop. The *italic* and **bold** phrases are inputs taken from the user. You have to display the required information in the same order and with the same words and characters as below. In order to better understand the given situation, some of the examples are shown on the chessboard.

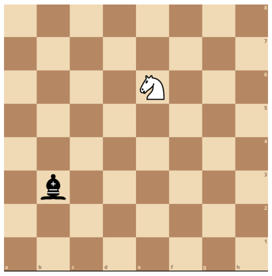
Sample Run 1

```
Please enter horizontal position of the knight (a,b,c,d,e,f,g,h): d
Please enter vertical position of the knight (1,2,3,4,5,6,7,8): 3
Please enter horizontal position of the bishop (a,b,c,d,e,f,g,h): c
Please enter vertical position of the bishop (1,2,3,4,5,6,7,8): 5
Knight can attack bishop
```



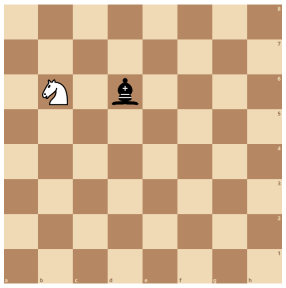
Sample Run 2

```
Please enter horizontal position of the knight (a,b,c,d,e,f,g,h): e
Please enter vertical position of the knight (1,2,3,4,5,6,7,8): 6
Please enter horizontal position of the bishop (a,b,c,d,e,f,g,h): b
Please enter vertical position of the bishop (1,2,3,4,5,6,7,8): 3
Bishop can attack knight
```



Sample Run 3

```
Please enter horizontal position of the knight (a,b,c,d,e,f,g,h): b
Please enter vertical position of the knight (1,2,3,4,5,6,7,8): 6
Please enter horizontal position of the bishop (a,b,c,d,e,f,g,h): d
Please enter vertical position of the bishop (1,2,3,4,5,6,7,8): 6
Neither of them can attack each other
```



Sample Run 4

```
Please enter horizontal position of the knight (a,b,c,d,e,f,g,h): h
Please enter vertical position of the knight (1,2,3,4,5,6,7,8): 2
Please enter horizontal position of the bishop (a,b,c,d,e,f,g,h): h
Please enter vertical position of the bishop (1,2,3,4,5,6,7,8): 2
They can't be in the same square
```

Sample Run 5

```
Please enter horizontal position of the knight (a,b,c,d,e,f,g,h): 7
Horizontal input for knight is not a letter
```

Sample Run 6

```
Please enter horizontal position of the knight (a,b,c,d,e,f,g,h): f
Please enter vertical position of the knight (1,2,3,4,5,6,7,8): a
Vertical input for knight is not a number
```

Sample Run 7

```
Please enter horizontal position of the knight (a,b,c,d,e,f,g,h): e
Please enter vertical position of the knight (1,2,3,4,5,6,7,8): 10
Vertical input for knight is not a proper number
```

Sample Run 8

```
Please enter horizontal position of the knight (a,b,c,d,e,f,g,h): A
Please enter vertical position of the knight (1,2,3,4,5,6,7,8): 4
Please enter horizontal position of the bishop (a,b,c,d,e,f,g,h): C
Please enter vertical position of the bishop (1,2,3,4,5,6,7,8): -
Vertical input for bishop is not a number
```

Sample Run 9

```
Please enter horizontal position of the knight (a,b,c,d,e,f,g,h): f
Please enter vertical position of the knight (1,2,3,4,5,6,7,8): 2
Please enter horizontal position of the bishop (a,b,c,d,e,f,g,h): ab
Horizontal input for bishop is not a letter
```

Sample Run 10

```
Please enter horizontal position of the knight (a,b,c,d,e,f,g,h): A7
Horizontal input for knight is not a letter
```

What and where to submit?

You should prepare (or at least test) your program using Python 3.x.x. We will use Python 3.x.x while testing your take-home exam. Let us repeat,

- You **must** use Google Colab to develop your code from scratch (from beginning till the end), and then submit it **through SUCourse+ only**! Once you are done with developing your code on Google Colab, then you will copy your code to CodeRunner to see if your program can produce the correct outputs. At the end, you will submit your code through CodeRunner (on SUCourse+). You should keep your Google Colab file until the end of the semester, we might want to look at this. ***If you fail to provide this Google Colab file anytime in the semester, you may not earn any credits from this Take Home Exam.***

- In CodeRunner, there are some visible and invisible (hidden) test cases. You will see your final grade (including hidden test cases) before submitting your code. Thus, it will be possible to know your THE grade before submitting your solution.

- There is no re-submission. You don't have to complete your task in one time, you can continue from where you left last time but you should not press submit before finalizing it. Therefore, you should make sure that it's your final solution version before you submit it.

General Take-Home Exam Rules

- Successful submission is one of the requirements of the take-home exam. If, for some reason, you cannot successfully submit your take-home exam and we cannot grade it, your grade will be 0.
- There is NO late submission. You need to submit your take-home exam before the deadline. Please be careful that SUCourse+ time and your computer time may have 1-2 minutes differences. You need to take this time difference into consideration.
- Do NOT submit your take-home exam via email or in hardcopy! SUCourse+ is the only way that you can submit your take-home exam.
- If your code does not work because of a syntax error, then we cannot grade it; and thus, your grade will be 0.
- Please submit your **own** work only. It is really easy to find "similar" programs!
- Plagiarism will not be tolerated. Please check our plagiarism policy given in the syllabus of the course.

Good luck!

Mehmet Kamil Ural
& IF100 Instructors