**IF100 – Fall 2021-2022**

**Take-Home Exam 4   
Due December 24th, 2021, Friday, 23:59 (Sharp Deadline)**

**Introduction**

The aim of this take-home exam is to practice on functions. You are given a main program that uses some functions and for the program to work correctly you should implement those functions. Therefore, you cannot finish this take-home exam without using functions.

For this Take-Home Exam, we share a colab file with you that only contains the main part of the assignment. Please do not modify the last code cell, instead use the previous code cell(s) to implement your own functions. If you modify the one given with the take-home exam documentation, then your grade will be affected with respect to the severity of the modification carried out. Therefore, do **not** modify the given main program. However, you still need to upload the entire solution (together with the main part) while submitting and checking your take-home exam with CodeRunner. That is, your submission should include the function implementations, and also the main program that is provided together with the take-home exam documentation.

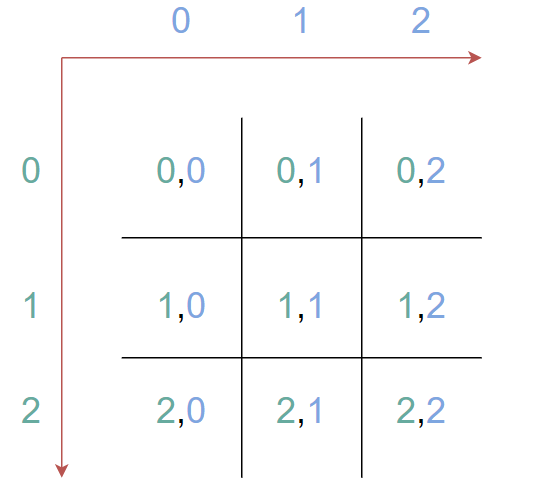
**Description**

In this take-home exam, you will be implementing a game called Tic-Tac-Toe. It is originally a paper and pencil game for two players who take turns marking the spaces in a three by three grid with X or O. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row is the winner. You will be writing the computer version of this game by using some functions which are expected from you to fulfill them. The descriptions of these functions are as follows:

* choose\_side() is a function to consult player1 which mark will be selected. This function does not take any parameter(s) and returns the chosen mark. If the user enters an invalid input which is anything other than "X" or "O", the same question will be asked to the user **until** the user enters a valid one. Notice that marks are **case sensitive**. The input message that you will take in this function is in the following format:

"Player1, please choose your side (either X or O): "

* display\_enviroment() is a function that outputs the three by three grid in its current state. It takes one parameter which is a list of lists (nested list) that contains the information of the locations of each mark and does not return anything. Notice that the elements of the list are the row locations and the elements of the inner list are the column locations. You can see the coordinates of each slot in the following figure:



The row edges of the board are constituted by using the following order: "-+-+-". For instance, if you call this function with the parameter [[" ", "X", "O"], ["O", " ", "X"], [" ", " ", " "]], then it should print the following output:



* makechoice() is a function that lets users choose a slot. You can assume that the user will always enter an input in the correct format. The input message that you will take in this function is in the following format:

"Player*player*, please make your choice: "

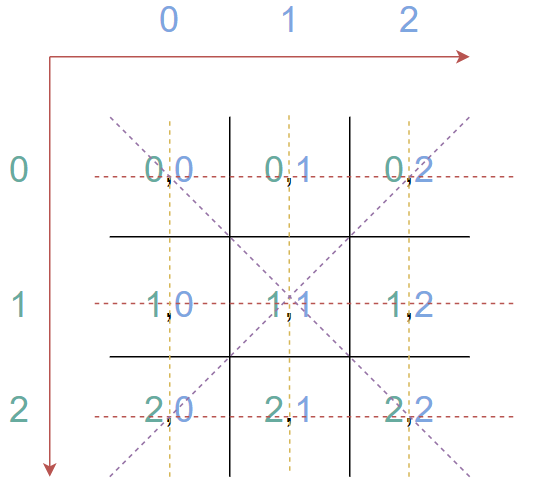
where *player* is the number of the user (so, it can be either "1" or "2"). Notice that you will take the *player* as a parameter which is in string type. This function will return a list where it contains only two elements which are row number and column number in order. Notice that these numbers can be in string or integer type. For example, if the user enters "1,2" as input, then this function should return either [1, 2] or ["1", "2"].

* implychoices() is a function that takes three parameters; nested list indicating the current board state, a list that contains the chosen row and col location respectively (this is the information you retrieved from makechoice() function), a string that indicates the player’s mark, and returns a nested list which is the new version of the board state. This function places a mark to the chosen slot (which is determined by using makechoice() function) and returns the updated nested list. However, you need to make sure that the chosen slot is available to mark. If not, then you need to print the following error line:

"*mark* is already in the location *rownum*,*colnum*. Skipping this turn."

Where *mark* is the mark of the player (so "X" or "O"), *rownum* is the location of the row, and *colnum* is the location of the column. If there is a mark in the chosen slot, then **the turn for that player will be skipped** which means that it will return the same board without any change.

* checkboard() is a function that checks if there is still an available slot to continue the game or not. It takes one parameter which is a nested list that indicates the current state of the board, and it returns True if there is an available slot. It returns False if all slots are marked.
* iswinner() is a function that checks whether any player won the game in the current state. It takes one parameter which is a nested list that indicates the current state of the board, and it returns the mark (either "X" or "O") of the winning player (if there is a winner). Otherwise (if there is no winner), it returns a space string (" "). The winning conditions are the following:  
  + placing three of their marks in a horizontal line is the winner.
  + placing three of their marks in a vertical line is the winner.
  + placing three of their marks in a diagonal line is the winner.



**In this take-home exam, you are given a main program, which aims at solving the above-described problem. This main program uses some functions, which are not provided. Your aim is to implement these functions, so that the program can work correctly.**

***You should not change the cell that contains the main program in the provided file. Therefore, any change on the given template main program may (and possibly will) result in grade reduction.***

**Apart from these 6 functions, please feel free to add any additional functions if you need. However, your code will not be working without implementing these mandatory 6 functions.**

You may check the Sample Run given below for further information.

**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.

***Sample Run 1***

Welcome to the Tic-Tac-Toe Game!

Player1, please choose your side (either X or O): **a**

Player1, please choose your side (either X or O): **x**

Player1, please choose your side (either X or O): **12**

Player1, please choose your side (either X or O): **O**

| |

-+-+-

| |

-+-+-

| |

Player1, please make your choice: **0,0**

O| |

-+-+-

| |

-+-+-

| |

Player2, please make your choice: **0,1**

O|X|

-+-+-

| |

-+-+-

| |

Player1, please make your choice: **1,0**

O|X|

-+-+-

O| |

-+-+-

| |

Player2, please make your choice: **2,2**

O|X|

-+-+-

O| |

-+-+-

| |X

Player1, please make your choice: **2,0**

O|X|

-+-+-

O| |

-+-+-

O| |X

Player1 has won the game!

***Sample Run 2***

Welcome to the Tic-Tac-Toe Game!

Player1, please choose your side (either X or O): **O**

| |

-+-+-

| |

-+-+-

| |

Player1, please make your choice: **1,1**

| |

-+-+-

|O|

-+-+-

| |

Player2, please make your choice: **0,1**

|X|

-+-+-

|O|

-+-+-

| |

Player1, please make your choice: **2,0**

|X|

-+-+-

|O|

-+-+-

O| |

Player2, please make your choice: **1,0**

|X|

-+-+-

X|O|

-+-+-

O| |

Player1, please make your choice: **1,2**

|X|

-+-+-

X|O|O

-+-+-

O| |

Player2, please make your choice: **0,2**

|X|X

-+-+-

X|O|O

-+-+-

O| |

Player1, please make your choice: **2,2**

|X|X

-+-+-

X|O|O

-+-+-

O| |O

Player2, please make your choice: **0,0**

X|X|X

-+-+-

X|O|O

-+-+-

O| |O

Player2 has won the game!

***Sample Run 3***

Welcome to the Tic-Tac-Toe Game!

Player1, please choose your side (either X or O): **X**

| |

-+-+-

| |

-+-+-

| |

Player1, please make your choice: **0,0**

X| |

-+-+-

| |

-+-+-

| |

Player2, please make your choice: **0,0**

X is already in the location 0,0. Skipping this turn.

X| |

-+-+-

| |

-+-+-

| |

Player1, please make your choice: **0,0**

X is already in the location 0,0. Skipping this turn.

X| |

-+-+-

| |

-+-+-

| |

Player2, please make your choice: **1,0**

X| |

-+-+-

O| |

-+-+-

| |

Player1, please make your choice: **1,1**

X| |

-+-+-

O|X|

-+-+-

| |

Player2, please make your choice: **1,0**

O is already in the location 1,0. Skipping this turn.

X| |

-+-+-

O|X|

-+-+-

| |

Player1, please make your choice: **2,2**

X| |

-+-+-

O|X|

-+-+-

| |X

Player1 has won the game!

***Sample Run 4***

*Welcome to the Tic-Tac-Toe Game!*

*Player1, please choose your side (either X or O):* ***X***

*| |*

*-+-+-*

*| |*

*-+-+-*

*| |*

*Player1, please make your choice:* ***0,0***

*X| |*

*-+-+-*

*| |*

*-+-+-*

*| |*

*Player2, please make your choice:* ***0,1***

*X|O|*

*-+-+-*

*| |*

*-+-+-*

*| |*

*Player1, please make your choice:* ***0,2***

*X|O|X*

*-+-+-*

*| |*

*-+-+-*

*| |*

*Player2, please make your choice:* ***1,1***

*X|O|X*

*-+-+-*

*|O|*

*-+-+-*

*| |*

*Player1, please make your choice:* ***2,1***

*X|O|X*

*-+-+-*

*|O|*

*-+-+-*

*|X|*

*Player2, please make your choice:* ***1,0***

*X|O|X*

*-+-+-*

*O|O|*

*-+-+-*

*|X|*

*Player1, please make your choice:* ***2,0***

*X|O|X*

*-+-+-*

*O|O|*

*-+-+-*

*X|X|*

*Player2, please make your choice:* ***2,2***

*X|O|X*

*-+-+-*

*O|O|*

*-+-+-*

*X|X|O*

*Player1, please make your choice:* ***1,2***

*X|O|X*

*-+-+-*

*O|O|X*

*-+-+-*

*X|X|O*

*The game is finished! It is a tie.*

**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 the CodeRunner to see if your program can produce the correct outputs. At the end, you will submit your code through CodeRunner (and 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 the 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!

Özgün Yargı

& IF100 Instructors