ASSIGNMENT 3

Read the instructions below carefully. The instructions must be followed. This assignment is worth 5% of your grade. The assignment is due on Monday 2nd of November 8AM. No late assignments will be accepted.

The goal of this assignment is to learn and practice (via programming and quizzes) many of the concepts that we have seen so far.

This assignment has 2 parts. Each part explains what needs to be submitted. Put all those required documents into a folder called a3_xxxxxx. Zip that folder and submit the zip file as explained in Lab 1. In particular, the folder should have the following files:

Part 1

a3_week2_xxxxxx.jpg, a3_week3_xxxxxx.jpg, a3_week4_xxxxxx.jpg, a3_week5_xxxxxx.jpg, a3_week6_xxxxxx.jpg

and

Part 2

a3_part2_xxxxxx.py

For any function that you design in part 2 of this assignment you have to include docstrings that specify:
- type contract
- description about what the function does
- preconditions, if any

****
PART 1 (5 points)
****

You can think of Part 1 as practice for your exams.

If you have difficulties with any of the tests below, I suggest strongly that you watch all the short coursera videos that relate to the given week. In general that is a good idea.

For Part 1 of the assignment 3 do the following:

Go to https://www.coursera.org/course/programming1

Click on Homeworks (on left hand side)

Complete Week 2, Week 3, Week 4, Week 5, and Week 6 exercises (by clicking Attempt Homework, like we did for Week 1 in one of the labs). For each Homework you have 5 tries. Once you are done go back to Exercise page. Click on Week 2 tab and close all the other tabs. Take a photo of the screen (do screen capture with your computer's software). What has to be visible in the photo is your name in the top right corner, the week of the test and the score of the last attempt. (See my example in file a3_week1_123456.jpg,
a3_week2_123456.jpg, a3_week3_123456.jpg and a3_week4_123456.jpg.

Do the same for Week 3, Week 4, Week 5, and Week 6.
Name each image as:

a3_week2_xxxxxx.jpg
a3_week3_xxxxxx.jpg
a3_week4_xxxxxx.jpg
a3_week5_xxxxxx.jpg
a3_week6_xxxxxx.jpg

You do not need to submit Week 1 score. We did that in one of the labs. I only included it in my images since that is the only test I completed, so you can see how your score and your name and the Week number should be visible. I have no score visible for
Week 2, 3, 4, 5 and 6 (because I did not do the tests). Yours will have to have "Last Attempted Score" for each of these, like mine does for week 1.

Make sure the following is visible in each of your images:
1. that your name is present on the top right corner
2. that the Week number is visible
3. that the score for the last attempt is visible for that week

important

Important: Each of your photos cannot be more than 200 KB (KB not MB!! we have limited space on the Blackboard Learn server). For me the screen capture produced a photo of about 100KB.

Each of the tests is worth 1 point. You will get full 5 points for Part 1 of the assignment, if all of your tests have 75% or more, otherwise the actual weighted average will be used.

****
PART 2 (45 points)
****
For this assignment, you should program a version of the memory game (to be described below). The game is usually played with cards. If you do not know what it is, google it or see and play a bit here, for example:

http://www.brainmetrix.com/memory-game/

Only your game will not use graphics libraries to display the board, but instead it will print the board with the tools that we have seen in class so far, i.e. printing on Python console.

As part of this assignment I provided the file called a3_part2_xxxxxx.py. All of your code has to god inside of that file (except that you will replace xxxxxx in the file name with your student number). The file already has some functions precoded for you.

Your game should use a (one dimensional) list as the board. The size of the board should be an even integer, size, between 2 and 52. This number should be obtained from the player. The board should be filled with the first size/2 capital letters of english alphabet such that each letter appears exactly twice in the list. And the order of these letters in the list should be random. As part of this assignment, I provided you with a function, called create_board, that does this for you.

You will see that a3_part2_xxxxxx.py has a function called play_game(board). That is where you should put your game paying code (notice the #YOUR CODE GOES HERE comment inside of that function) The function play_game takes the board as input parameter (the board created by the function create_board mentioned above). Once the player completes the game, you should also print the number of guesses it took to solve it and how far it is from the optimal (although impossible without luck) size/2 guesses.

Outside of that function you will obtain the desired size of the board from the player (in the space indicated by #YOUR CODE GOES HERE TOO)

More about the game:

When your program prints the board, the locations for which paring is not discovered yet should display * and the locations for which the paring is discovered should display the letter that is on that location. In addition under the board each location should be labeled from 1 to size to help the user identify which locations they want opened next.

the letters and location number have to be nicely lined up

You may assume that the player will follow your instructions and will input integers (rather than strings say), but your program should test if the player entered integers
in the required range and prompt her to repeat the entry until correct input is obtained. You should also fully test your program -- for example, what does your program do if the player does something silly, like entered two locations that are already discovered, etc ..

Here is what a run of your program should look like. Study the below example carefully to understand what your program should do. Among other things, you will see that your program will need to ``clear screen'' (I explain why is this needed more at the end of this file). It is ok to implement that by, for example, simply printing 30 or so new lines. You will also see that you will need to pause the execution of the program so that the player can look at the newly opened two elements. Once the player has observed the two new elements ~~that~~ she should presse enter and the program will continue (you can assume here that she follows instructions). To have your program pause and wait for the player to press enter use the provided function called wait_for_player()

then

see function called print_board

Also, think of the design of your program. For example, your program should have a function that displays the current board (I provided that), but it should also have a function that displays the board with two new positions revealed (did not provide that). These functions would be called from your game playing function, i.e from play-game

Designing your program by decomposing it into smaller subproblems (to be implemented as functions) makes programming easer, less prone to errors and makes your code more readable. You will be graded on these aspects of your program too.

Here is finally what a run of your program should look like:

```
Your program:
How many cards do you want to play with?
Enter an even number between 2 and 52:

Player: 5

Your program:
How many cards do you want to play with?
Enter an even number between 2 and 52:

Player: 6

Your program:

*    *    *    *    *    *
1    2    3    4    5    6

Enter two distinct locations on the board that you want revealed.
i.e two integers in the range [1, 6]

Player: 1
        2

Your program:

  C    B    *    *    *    *
  1    2    3    4    5    6
Press enter to continue

Player: presses enter

Your program: clears the screen (so that the above board is not
visible) and prints

*    *    *    *    *    *
1    2    3    4    5    6

Enter two distinct locations on the board that you want revealed.
```

```
i.e two integers in the range [1, 6]

Player: 3
        3

Your program:

Enter two distinct locations on the board that you want revealed.
i.e two integers in the range [1, 6]

Player: 3
        4

Your program:
*    *    A    C    *    *
1    2    3    4    5    6
Press enter to continue

Player: presses enter

Your program: clears the screen and prints

*    *    *    *    *    *
1    2    3    4    5    6

Enter two distinct locations of the board that you want revealed.
i.e two integers in the range [1, 6]

Player: 1
         4

Your program:

C    *    *    C    *    *
1    2    3    4    5    6
Press enter to continue

Player: presses enter


Your program: clears the screen and prints

C    *    *    C    *    *
1    2    3    4    5    6
Enter two distinct locations of the board that you want revealed.
i.e two integers in the range [1, 6]


Player: 1
         4
```

Your program:

```
C   *   *   C   *   *
1   2   3   4   5   6
Press enter to continue
```

Player: presses enter

Your program: clears the screen and prints

```
C   *   *   C   *   *
1   2   3   4   5   6
Enter two distinct locations of the board that you want revealed.
i.e two integers in the range [1, 6]
```

```
Player: 2
        5
```

Your program:

```
C   B   *   C   A   *
1   2   3   4   5   6
Press enter to continue
```

Player: presses enter

Your program: clears the screen and prints

```
C   *   *   C   *   *
1   2   3   4   5   6
Enter two distinct locations of the board that you want revealed.
i.e two integers in the range [1, 6]
```

```
Player: 3
        5
```

Your program:

```
C   *   A   C   A   *
1   2   3   4   5   6
Press enter to continue
```

Player: presses enter

Your program: clears the screen and prints

```
  C   *   A   C   A   *
  1   2   3   4   5   6
Enter two distinct locations of the board that you want revealed.
i.e two integers in the range [1, 6]
```

Player: 2

```
        6

C   B   A   C   A   B
1   2   3   4   5   6
Press enter to continue
```

Player: presses enter

Your program prints

```
C   B   A   C   A   B
1   2   3   4   5   6
```

Congratulations! You completed the game with 7 guesses.
That is 4 more than the best possible.

```
==============================================================
an extra note:
==============================================================
```

Here is a brief clarification about why you need to be able to "clear"
the screen and pause the game:

For example, let's say your program just printed the current board.

```
B  *   *  B  *  *

1  2   3  4  5  6
```

Then it asks the player to enter two new locations, and player enters 2 and 3

Then your program will print:

```
B  A   C  B  *  *

1  2   3  4  5  6
```

It is here that you need to pause the program. After the player
observed the two new locations, the next thing that needs to be done
is to clear the screen (meaning the above thing should not be visible anymore) and print the
board again. Which would print

```
B  *   *  B  *  *

1  2   3  4  5  6
```

If you do not pause where I stated above, the computer will do the above so fast that the player will only see

```
B  *   *  B  *  *

1  2   3  4  5  6
```

Then she will be asked to enter two new locations, but after she enters them she will only get to see this again (since the board with two new locatins is already gone from the screen)

```
B  *   *  B  *  *

1  2   3  4  5  6
```

As for clearing the screen, you can do that by e.g. calling print() about 30 or more times