ENGR 102 – Fall 2021
Lab Assignment #7

Deliverables:

There are two deliverables for this team assignment. Please submit the following files to Mimir:

- `Lab7a_Act1.pdf`
- `Lab7a_Act1.py`

Activity #1: *Checkers Moves– Team Activity* **(https://en.wikipedia.org/wiki/English_draughts)**

The purpose of this activity is to get you used to using lists of lists, in a 2-D matrix-like format. Your team will create a program that sets up a standard (English) checkers board and lets users make moves.

First, **BEFORE YOU CODE**, you must **think** about how to structure your program. Create a document named Lab7a_Act1.pdf that contains an algorithm (in English or pseudocode, **NOT** python) of how you want your program to work. You may want to have several smaller algorithms that do one task each (think functions) instead of one large algorithm.

Here are the details:

- The board is an 8 x 8 board. **Use a list of lists** to store your current board.
- Display the current state of the board before every move. Each empty square should just be a period. Each square with a piece should have that piece's identifier.
- For identifiers, you may use lower-case for the light pieces and upper-case for the dark pieces.
  - Use O/o or distinct round symbols (*, @, ●, etc).
  - For a fun alternative, see the note at the bottom.
- When a piece is moved, the location it moves from is then an empty square, and the piece then occupies the square it is moving to. If there was already a piece in the square it is moving to, then that other piece is removed from the game.
- Continue to let users move pieces until they enter "stop".
- **Important:** You do **NOT** need to enforce any rules of checkers or verify moves, with two exceptions:
  - If a user tries to move from a position where there is not a piece, print a message to the screen and try again.
  - Only allow users to move to dark squares (move diagonally). If a user tries to move to a light square, print a message to the screen and try again.
  - Other than that, you don't need to worry about the sides alternating turns, pieces moving in ways they aren't allowed, landing on your own pieces, jumping over pieces, crowning pieces, etc.
  - Only one piece moves at a time, and it can move to any dark square on the board.

You need to specify the system you want to use to specify the starting/ending locations on the board.

- In your pdf document, include instructions for user input.
- This problem will be graded manually so **your instructions must be clear**.
- In checkers a standard system is used where dark squares are numbered 1 through 32 starting at the top left and continuing right and down. You do not have to use this system, but you can if you wish.

Your pdf must include an algorithm and instructions for running your program.

Now that your team has planned everything, write a program named Lab7a_Act1.py that sets up a checkers board and lets users make moves. **Remember to write test cases and use the "pyramid" approach to programming!**

As an example, here is what the board would look like at the very beginning of the game:

```
.o.o.o.o
o.o.o.o.
.o.o.o.o
........
........
o.o.o.o.
.o.o.o.o
o.o.o.o.
```

Remember to discuss this as a team and *think* through exactly what you will do *before* developing it!

Note: You can display empty and filled circles using Unicode characters like the example below.

```
print(chr(9675))    # displays empty circle: ○
print(chr(9679))    # displays filled circle: ●
```

You can also change the color of your output using the `colorama` package like the example below.

```
from colorama import Fore, Style
print(Fore.RED + Style.BRIGHT + 'my text' + Fore.RESET + Style.NORMAL)
# displays 'my text' in bright red, then resets future text back
# to default colors
```