**Specification for Assignment 8 of 8**

*Your submission* ***must include a completed copy of the assignment header*** *that* ***must include your name*** *(as it appears on cuLearn),* ***student number****, and* ***section****.*

*Your submission* ***must be a source file named 'comp1405_f17_#########_a8.py'*** *(with the number signs replaced by your nine digit student number),* ***must be written*** *using the* ***Python 3*** *language, and* ***must run on the official virtual machine****.*

***If your program does not use the pygame library then do not compress*** *your submission. However,* ***if you choose to have program draw the chessboard using pygame*** *then you are* ***permitted to compress your submission*** *into a "zip" file in order to allow you to include graphical assets (e.g., https://opengameart.org/content/chess-pieces) with your submission.*

***Your submission must then be named 'comp1405_f17_#########_a8.zip'.***

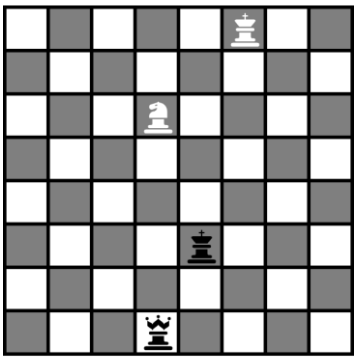
***Late assignments will not be accepted*** *and will* ***receive a mark of 0****.*

***Submissions that crash*** *(i.e., terminate with an error) will* ***receive a mark of 0****.*

***The due date for this assignment is Thursday, November 30, 2017, by 11:30pm.***

For this assignment, you will write a program that can **model a chess game in progress** and will **assess (numerically) who is winning** using the relative value system. You can read more about this approach at http://en.wikipedia.org/wiki/Chess_piece_relative_value, but please note that **you must use the values specified below**.

| Type | Value |
|---|---|
| King | 0 |
| Queen | 10 |
| Rook | 5 |
| Knight | 3.5 |
| Bishop | 3 |
| Pawn | 1 |



According to the relative value system,
"White" currently has a score of 3.5 and "Black" currently has a score of 10.

Therefore "Black" is currently winning.

To accomplish this task your program must:

- **Provide instructions** to the user for using your program,
- Provide a way for the user **to type in the current state of a chessboard**,
- **Calculate the score** associated with each player and report which player is winning,
- Provide a way for the user **to move pieces**, and
- **Print** (**or draw** using pygame) the **current state** of the chessboard.

Further details about how to complete these tasks are specified below.

To allow the user to type in the current state of a chessboard, you must ask the user to type what pieces appear on each of the eight rows separately. Your program must use lowercase letters for the white pieces and uppercase letters for the black pieces, and your program must use the hyphen "-" for an empty space and the following abbreviations – (K)ing, (Q)ueen, (B)ishop, k(N)ight, (R)ook, and (P)awn. If the user tries to enter a row that contains an invalid character or is more or less than eight characters in length, your program must reject that row and have them retype it.

As a clarifying example, to submit the chessboard on the previous page the user would be expected to type the following:

```
-----k--
--------
---n----
--------
--------
----K---
--------
---Q----
```

**Your program must store the state of the chessboard as a two-dimensional list**, but it is up to you whether you would like to store the board as a list of lists of characters or process it into a list of lists of integers instead. Once you have loaded the board you must present it to the user, either by printing it to the terminal or by drawing it to a pygame window.

Since **your program must store the value associated with each piece** (from the table on the previous page) **in a list of floating-point values**, you must then find a way to compute the sum of the values corresponding to the pieces of each player. Since your chessboard must be stored as a multidimensional list, **you must use nested loops** to accomplish this.

**You must give the user the option to quit** the program or to either **enter another chessboard from scratch** or simply **move a piece**. It is not necessary for your program to determine whether or not a move is legal, so you can accomplish this by asking the user the location of the piece to be moved and, if there is a piece at that location, ask the user for the location to which to move that piece. Once the piece has been moved your program is expected to recompute the scores and present the user with their options again. **Your program should repeat these steps until the user selects quit.**

Please note that **you are not creating a program for playing chess** with this assignment. Instead what you are creating is a program for providing extra information about existing chess games - your program operates inside a loop that gives the user the option of either entering the complete chessboard for a game in progress or moving a piece in the chessboard currently loaded in memory.

**If the player chooses to move** a piece, your program must **ask them to specify a row and column**. **If they specify an invalid position or there is no piece** at that location then your program will need to **respond to this error**, but **if the specified location is on the chessboard and contains a piece** your program will **ask for another location**. Your program is then expected to **move that piece to the new location, overwriting whatever was there before**.

(Re)consider, as a clarifying example, this chessboard included earlier, this time presented with labels on the row and column *(n.b., it is not necessary for you to use the same labels I have used here, as long as the user knows what you are expecting)*:

```
  12345678
A -----k--
B --------
C ---n----
D --------
E --------
F ----K---
G --------
H ---Q----
```

If the user chooses to move a piece and selects row F and column 5 (i.e., the black king) and then chooses to move that piece to row C and column 4, the board would become:

```
  12345678
A -----k--
B --------
C ---K----
D --------
E --------
F --------
G --------
H ---Q----
```

After this move, the scores for the black and white players would be 10 and 0, respectively. *(n.b., It doesn't matter that the move from F5 to C4 isn't legal for a king, because as noted previously you aren't playing a game of chess - you are just allowing the user to explore a game in progress.)*

**Specification for Assignment 8 of 8**

If you elect to draw the state of your chessboard using pygame then **your solution may import the pygame library** but your solution **must not import any other libraries.**

Your solution **must not use list comprehensions** in any way.

Your solution **must not use dictionaries or maps** in any way either.

You are **not permitted to use any built-in functions except input, print, range, len, insert, append, pop, ord, chr, type, and the functions for creating or drawing upon a pygame window.**

Please note that 'in' and '+' are operators, not functions, so you are free to use these if you wish. You may also use the indexing and slicing operators.

You must **include a completed header** with your name and student number.

**You must write a main function that takes no arguments and produces no return values.** Additionally, as the very last line of your program, you must call this main function.

Your program must be **completely organized into a collection of functions**, and your program is **not permitted to use any global variables** whatsoever. Including the main function mentioned above, your program must be organized into at least four functions (but you may add as many as you wish).

You must **use meaningful variable names** and you must **comment your code**.

In addition to commenting your code normally, y**ou are also required to write a descriptive comment for each of the functions you create for this assignment**. This comment must begin with a short description of what the function does, contain a list (labeled @params) of indented items corresponding to the parameter variables you use to hold your function arguments, and contain a description of any return values (labeled @return). If the function you are writing has no params or returns, then you must still include the @params or @return tag and write (and indent) the word none.

**Failure to observe any of these requirements will result in penalties.**

---

Please remember that **this assignment must be completed individually**. **Do NOT discuss this assignment with anyone** except the instructor or the teaching assistants, and **do NOT look for code samples on the internet** or from any other source.

As with the previous assignment, **you must create flowcharts** before you start writing your code for this assignment. You are expected to create a flowchart for each of the functions you will be writing. You will not be submitting any flowcharts on cuLearn, but if you visit the instructor or a teaching assistant **you will be asked to show your flowcharts before you will receive any assistance**.