# University of Western Ontario, Computer Science Department
# CS1026A SU19, Computer Organization

**Assignment 4** **Due: June 7, 2019**

---

**General Instructions:** This assignment consists of 9 pages, 1 exercise, and is marked out of 100. Assignments are the independent work of each student. Software may be used to detect cheating.

**Non-Functional Code Instructions:**

1. Include brief comments in your code. Identifying yourself (the code's author) by name and user ID in the initial comment header. Comment also on key instructions and calculations in your code.
   e.g.
   ```
   ##
   # Plays the game of tic-tac-toe.
   # Student Name:  Alex Brandt
   # Student ID: abrandt5
   ```
2. Follow *good coding style* and normal Python conventions. This includes, but is not limited to:
   (i) Meaningful variable names.
   (ii) Conventions for naming variables and constants.
   (iii) Use of constants over "magic numbers".
   (iv) *Readability*: indentation, appropriate white space (blank spaces) within instructions, consistency in the use of all of the above.

**Evaluation:**

1. Functional Requirements:
   (i) Does your classes correctly implement its specification?
   (ii) Does your main program behave according to the specifications?
   (iii) Does your main program handle invalid input?
   (iv) Does your main program output everything as expected?
2. Non-Functional Requirements (above).
3. Ability to follow directions precisely.

**Submission Instructions:** Your submission should include exactly two files (zipped into a zip file, if you'd like). These two files are: `myBoards.py` and `userid_tictactoe.py` where userid is replaced by your UWO User ID (everything preceding "@" in your UWO email; e.g. abrandt5). The contents of these two files are described below.

**Learning Outcomes:** In this assignment we will look at using classes, objects, and saved states, and program interaction.

**Exercise 1.** In this assignment you will implement a simple text-based game of tic-tac-toe (a.k.a. naughts and crosses, X's and O's), see https://en.wikipedia.org/wiki/Tic-tac-toe. To complete such a task, you should implement two classes and two Python files. The first file, `myBoards.py`, will hold two classes `GameBoard` and `ScoreBoard`. The second file, `userid_tictactoe.py` will hold your `main` function and effectively run the loop which drives progress through the game.

**Part 1.** In this section we will define the contents of the `myBoards.py` file. In this file we look to implement two classes, `GameBoard` and `ScoreBoard`.
The class **`GameBoard`** :

- Holds a representation of the game board throughout the running of a single game of tic-tac-toe.
- You should use a 3-by-3 table to encode the game board. You may use any data you wish inside the table, but I recommend strings.
- Must have a `printCurrentBoard` method which prints the current state of the game board in a nice way. That is, it should print the board itself as well as any "X" or "O" that have been placed so far on the board by the players. For example:

```
X| |
-----
 | |O
-----
O| |X
```

- Must have a `placeX` method which adds an "X" to some space on the board. This method takes `self` (as all Class methods take) and two additional parameters `i` and `j` which are the row index and column index, respectively, of where to place the "X" on the board. This method returns True if the move is legal (i.e. the space specified is blank) and False otherwise. If the move is legal, the table should be updated to indicate an "X" is now in that space on the board.
- Must have a `placeO` method which adds an "O" to some space on the board. This method takes `self` and two additional parameters `i` and `j` which are the row index and column index, respectively, of where to place the "O" on the board. This method returns True if the move is legal and False otherwise. If the move is legal, the table should be updated to indicate an "O" is now in that space on the board.
- Must have a `decideWinner` method which determines if there is a winner on the current board. It should determine if "X" is the winner or "O" is the winner. If neither are winners then it should return a special value to indicate there are (yet) no winners.
- Must have a `boardFull` method which returns a Boolean value to indicate if the board is full (and thus no more legal moves exist and the game is over). This method returns True if the board is full and False otherwise.

The class **ScoreBoard** :

- Holds a representation of a score board.
- Must keep track of the number of wins, number of losses, and number of draws that have occurred for each player since the start of the program (see part 2 for further explanation on a "player").
  - *Hint:* use a few dictionaries or a dictionary of lists to encode this data.
- Must have an `addWin` method which takes as parameters `self` and a player's name. This method increases the input player's win count by 1.
- Must have an `addLoss` method which takes as parameters `self` and a player's name. This method increases the input player's loss count by 1.
- Must have an `addDraw` method which takes as parameters `self` and a player's name. This method increases the input player's draw count by 1.
- Must have a `printScoreBoard` method which **cleanly** prints the name of each player and their number of wins, losses, and draws **as a table**. For example:

```
          Name | Wins | Losses | Draws
---------------|------|--------|-------
Foo            |    1 |      0 |     1
Bar            |    0 |      1 |     1
```

  this
  - *Hint:* use `%ns` where `n` is a number to help with .

**Part 2.** In this section we define the contents of the `userid_tictactoe.py` file. In this file you shall `import` your other file (e.g. by the command `from myBoards import *`) and then use those imported classes within your `main` function. Your main function will run the *game loop* which, in general, follows the following format:

(i) Take input from the user

(ii) Update the current game state

(iii) Output new information to the user

Specifically, your `main` function should do the following.

1. Begin by printing out instructions to the user on how to play your game. For example,

```
We are playing tic tac toe!
To play the game enter two numbers to indicate where to place
each game piece.  Enter numbers from 1 to 3.
(1,1) is the top left corner.  (3,3) is the bottom right corner.
```

2. Ask the users to enter the names of the two players who are playing the current game. Ask who will play as X's. Then ask who will play as O's. A player's name is their identifier for the `ScoreBoard`.

3. Then enter the game loop, which repeats until a winner is found or a draw occurs. The game loop is:

   (i) Print out the state of current game board.

   (ii) Tell the players whose turn it is *by their name*. For example, "It's Alex's Turn!".

   (iii) Prompt the current player to enter their move as two numbers on one line. For example, "Where should X go?"

   (iv) Parse the line entered by the player (e.g. by using `split` to get the individual numbers) and use either the `placeX` or `placeO` methods of the `GameBoard`, as appropriate, to update the board's state. If their move is invalid (either outside the bounds of the board or is on top of another piece which is already played) go back to step (iii).

   (v) Determine if there is a winner by using `GameBoard`'s `decideWinner` method. If there is a winner, tell the players who the winner is, print the final game board, and exit the game loop.

   (vi) Determine if there is a draw by using `GameBoards`'s `boardFull` method. If there is a draw, inform the players of the draw, print the final game board, and exit the game loop.

4. Update the scoreboard for the current players based on wins/losses/draws and print out the current scoreboard. You should make use of the `ScoreBoard`'s `addWin`, `addLoss`, and `addDraw` methods as appropriate.

   - *Note:* the printed scoreboard should contain *all players* from the current execution of the program. Therefore, if you play again and the players' names change, then your score board should have more than two entries in it. The order of entries printed does not matter.

5. Ask the user if they want to play again. If they respond with any string containing "y" or "Y", then go back to step 2 and play the game again. Otherwise, terminate the program.

**An Example Playthrough:**

```
We are playing tic tac toe!
To play the game enter two numbers to indicate where to place each game piece.
Enter numbers from 1 to 3.
(1,1) is the top left corner. (3,3) is the bottom right corner.

Who is playing as X? Foo
Who is playing as O? Bar

 | |
-----
 | |
-----
 | |

Foo's turn! Where should X go? 1 1

X| |
-----
 | |
-----
 | |

Bar's turn! Where should O go? 2 3

X| |
-----
 | |O
-----
 | |

Foo's turn! Where should X go? 1 2

X|X|
-----
 | |O
-----
 | |

Bar's turn! Where should O go? 2 2

X|X|
-----
 |O|O
-----
```

```
 | |

Foo's turn! Where should X go? 1 3
Winner is X !

X|X|X
-----
 |O|O
-----
 | |

          Name | Wins | Losses | Draws
---------------|------|--------|-------
Foo            |    1 |      0 |     0
Bar            |    0 |      1 |     0

Play again? (Y/N) y
Who is playing as X? Bar
Who is playing as O? Buzz

 | |
-----
 | |
-----
 | |

Bar's turn! Where should X go? 1 1

X| |
-----
 | |
-----
 | |

Buzz's turn! Where should O go? 2 2

X| |
-----
 |O|
-----
 | |

Bar's turn! Where should X go? 3 3

X| |
-----
```

```
 |O|
-----
 | |X

Buzz’s turn! Where should O go? 3 2

X| |
-----
 |O|
-----
 |O|X

Bar’s turn! Where should X go? 1 2

X|X|
-----
 |O|
-----
 |O|X

Buzz’s turn! Where should O go? 1 3

X|X|O
-----
 |O|
-----
 |O|X

Bar’s turn! Where should X go? 3 1

X|X|O
-----
 |O|
-----
X|O|X

Buzz’s turn! Where should O go? 2 1

X|X|O
-----
O|O|
-----
X|O|X

Bar’s turn! Where should X go? 2 2
That tile is already occupied!
```

```
X|X|O
-----
O|O|
-----
X|O|X

Bar’s turn! Where should X go? 2 3
That game ended in a draw!

X|X|O
-----
O|O|X
-----
X|O|X

          Name | Wins | Losses | Draws
---------------|------|--------|-------
Foo            |    1 |      0 |     0
Bar            |    0 |      1 |     1
Buzz           |    0 |      0 |     1

Play again? (Y/N) y
Who is playing as X? Foo
Who is playing as O? Bar

 | |
-----
 | |
-----
 | |

Foo’s turn! Where should X go? 1 1

X| |
-----
 | |
-----
 | |

Bar’s turn! Where should O go? 2 2

X| |
-----
 |O|
-----
```

```
 | |

Foo's turn! Where should X go? 1 2

X|X|
-----
 |O|
-----
 | |

Bar's turn! Where should O go? 3 3

X|X|
-----
 |O|
-----
 | |O

Foo's turn! Where should X go? 1 3
Winner is X !

X|X|X
-----
 |O|
-----
 | |O

          Name | Wins | Losses | Draws
---------------|------|--------|-------
Foo            |    2 |      0 |     0
Bar            |    0 |      2 |     1
Buzz           |    0 |      0 |     1

Play again? (Y/N) n
```