The goal is to build a one-player Connect-4 game, so that a human player can play the Connect-4 (or 4-in-a-line) against your program. For the first game of the match, the human player will always go first. Next, your program will display the current game board, a slot that is not filled will be marked by “-” and a slot filled by human as “X” and a slot filled by the program as “O”. Next, your program will ask for an input from the human player, detect whether the move is valid, and then whether the move leads to a win for the human player. If not, decide your next move and detect whether it leads to a win, then go back to poll the human player. After each slot is filled, your program will display the game board. When a win is detected, the program pauses, display that either the human or the computer has won. At the end of each game, your program will poll the human player to see whether he/she wants to continue. If the answer is yes, the order of the player will switch (if previously the human went first, now the computer will go first).

Ultimately, you will build an intelligent PC vs. Human Connect-4 game. To do so, we break the task into 3 parts. Submit your program via turnin (make sure to include detailed comments for each instruction, otherwise we will deduct 50% of the point). The weight of these three labs are equal (each contributes 30% of the total lab score).

**Lab 1:** you will construct the input/output process where you collect moves from the human players, determine moves for the program, and display the game board, until all the slots are filled (DRAW). The move of the human player depends on its input (i.e. the column it selects). The program’s move will be the first available slot of the lowest unfilled column. For example, if the column 1 is not occupied at all, then the program will occupy column 1, row 6. If the column 1 is occupied till row 3, then the program will occupy column 1, row 2. If column 1 is completely occupied, then the program will move to column 2.

**Note:** If a human’s move tries to fill an occupied slot, you will print out a warning and ask for a new move. And your program’s move must not fill an occupied slot at all.

**Due:** **Friday 4/29 11:59PM** via Turnin, use **lab1** in your turnin command;

- **Lab 2:** you will detect whether the human or the program has won the game after each move is made. But no need to get the program to make smart move to block or win, just use the same strategy in Lab1. The result of the game will be either WIN or DRAW.

**Due:** **Friday 5/13 11:59PM** via Turnin, use **lab2** in your turnin command;

- **Lab 3:** you will replace the dumb strategy with a set of customized strategies to maximize the program’s winning. In addition to the code, you will describe in one-page text the strategies used to maximize the winning.

**Due:** **Friday 5/20 11:59PM** via Turnin, use **lab3** in your turnin command;
**Program Guide:**

You have a game board of 6x7, a total of 42 slots (like the one from this link [http://www.mathsisfun.com/games/connect4.html](http://www.mathsisfun.com/games/connect4.html)).

When you print the game board, make sure it follows a 6x7 grid:
(for example: X be the human and 0 be the computer)

```
  1  2  3  4  5  6  7
1 - - - - - -
2 - - - - - -
3 - - - - - -
4 - - - - - -
5  O  O  X  X  -  -
6  X  X  O  O  X  O  X
```

[REQUIRED]: To store the game board, you will create an array of 42 elements:

```
Gameboard: .space 168
```

[REQUIRED]: At the start of each game, you will **set each array element to 0**, indicating that all the slots are unoccupied. When the human occupies a slot, you change the corresponding array element value to 1, and when the computer occupies the slot, you set the array element value to 2.

For a human to input his slot selection, he/she will indicate by the column number, e.g. 7 means column 7; after getting this number, you will set the largest unfilled row in this column as the input location of the human player.