Today in lab we will build pieces of what might go into a Five in a Row type game. Look at the link GoBoardDiagrams.pptx on the webpage to see some structure diagrams.

I. Intro: Our first task is to build a piece of the board, with a marker indicating that a player has placed a stone. We will use W for white, and B for black.

Here is what we eventually would like to produce, one of these:

```
W----     ----W     ----     ----
|   |     |   |     |   |     |   |
-----     ----     W----     ----W
```

A) **Write four functions** (you pick the names - good ones), to produce the following.

First function:

```
-----
```

Second function:

```
W----
```

Third function:

```
----W
```

Fourth function:

```
|   |
|   |
```

Write a **main** to test these four functions. (See first structure diagram.)

B) **Modify main** to draw each of the four boxes, one under another, separated by a blank line, using calls to these four functions (and other print statements for the blank lines).

C) **Modify the second and third functions** to take an integer parameter **turn**. (See second structure diagram.) If the value of **turn** is 1, they should print a W, as above. But if the value of **turn** is -1, they should print a B in place of the W. (One player plays black and the other plays white. Whose **turn** is it?)

Modify **main** to have an int variable **player**, and use it as the argument to the second and third function in all four boxes.

```
player (argument in main) ← turn (parameter in the function)
```

**Test main** again, using both -1 and 1 as values of **player**.

D) At the beginning of **main** set **player** equal to 1.

In **main**, after the chunk of code that draws the first box, **insert the line**

```
player = player * -1;
```

Insert this same line after the chunk of code that draws the second box and the third box. Run the program. **What happens to the boxes you see?**
E) In the second and third functions you may have some lines that look something like

```
    print("W----");
```

and others with a B in the same place as the W. In both of these functions introduce a
local string variable `color` and set it to 'W' or 'B' depending on the value of `turn`. Use an `if...else` to set the value of color. Then use **only one print statement**,
```
    print("{0}----".format( color));
```
completely after the `if...else`, not inside it. (You will need a slightly different
print statement in the third function.)
Run `main` and see that everything still works.

F) `Main` is not the only function that can call other functions. Create a new function, `box(player)`

This function's job, for now, is to draw all four boxes that `main` used to draw. (But all for
the same `player`. You will no longer need `player=player*-1.`) (See third structure
diagram.) Main should no longer draw the boxes itself, but instead should invoke `box` to
do it.

You can move some of the code from `main` into this function. Think about which lines
of code from main were needed for this task and move them into `box`.

Modify `main` to call this function `box` (instead of drawing the boxes itself), and test it.

G) In truth, we only want one of the four boxes drawn by `box` (at any function call). (See
fourth structure diagram.) Add a new parameter to `box`,

```
    box(player, corner)
```

where `corner` can have the values 'A', 'B', 'C', or 'D'. You will modify `box` to print
only one of the four boxes, depending on the value of `corner`:

- for 'A' the letter (W or B) appears in the upper left corner.
- for 'B' the letter (W or B) appears in the upper right corner.
- for 'C' the letter (W or B) appears in the lower left corner.
- for 'D' the letter (W or B) appears in the lower right corner.

You should use `if, elif, and else` statements so that it only checks whether
`corner` is 'B' if it has found out that it is not 'A', and so on. You will also want to indent
the lines of code for each option.

Modify `main` to call this newer version of `box`. Each call to `box` should produce only
one square. Have `main` call `box` many times, to test all possibilities. To be sure that
each call is only drawing one square, try a different sequence, perhaps using 'D', then 'B',
then 'A', then 'C' and look at your output to make sure the correct sequence of squares is
drawn. Changing the sequence should only involve changes to `main`, not to `box`.

II. Submitting your work.
• Make sure your name is on this lab packet.
• Staple
  your program file, with the output to the Lab 13 packet.
• Turn the lab in at the file cabinet.