CS 1110, LAB 10: BLACKJACK

http://www.cs.cornell.edu/courses/cs1110/2017sp/labs/lab10.pdf

First Name: ____________ **Last Name**: ____________ **NetID**: ____________

Getting Credit: Deadline: **in the first 10 minutes of (your) lab next week (Tues Apr 18/Wed Apr 19)**. The checking-off procedure is the same as before.[1]

As usual, create a new directory on your hard drive for this lab's files. Then, download into that new directory the files you need for lab 10; get them packaged in a single zip file from the Labs section of the course web page, http://www.cs.cornell.edu/courses/cs1110/2017sp/labs .

1. The Game of Blackjack

In this lab, you will finish a class definition for `Blackjack` that a casino could use to run multiple blackjack games simultaneously.

A player wins at blackjack by ending with a hand that has more points than the dealer's, but not more than 21 points. If someone exceeds 21 points, they are said to have "gone bust" and immediately lose. Points come from the ranks of the cards in a hand: 10 points for each face card (Jack, Queen, or King), 11 points for an ace, and the rank of the card for anything else (e.g., a 4 of anything is 4 points).[2]

Play begins with two cards being dealt to the player and one card to the dealer. All cards in each hand are always visible to all participants. The player can chose to "hit" (get an additional card from the deck) or "stay" (turn over play to the dealer). If the player eventually stays without going bust, then the dealer draws cards until they go bust or decides to stop.

Once you complete the lab, you can relax and play a few rounds of the game yourself. The next page shows a sample transcript from our solution code.

Course authors: E. Andersen, D. Gries, L. Lee, S. Marschner, C. Van Loan, W. White

[1] In case you've forgotten, here's a reminder: Show this handout and/or your code to a staff member either (a) during your lab 10 session, (b) in non-professorial consulting hours listed at http://www.cs.cornell.edu/courses/cs1110/2017sp/about/staff.php up to the day **before** your next scheduled lab section, or (c) in the first 10 minutes of (your) next scheduled lab (Tues Apr 18/Wed Apr 19). Beyond that time, the staff have been instructed not to give you credit.
Labs are graded on effort, not correctness. We just want to see that you tried all the exercises, and to clarify any misunderstandings or questions you have.

[2] In some versions of blackjack, an ace can be worth either 1 or 11, whichever is better; we will ignore that rule for our implementation.

```
[llee: lab10] python blackjack.py
Welcome to CS 1110 Blackjack.
Rules: Face cards are 10 points. Aces are 11 points.
       All other cards are at face value.

Your hand:
8 of Spades
6 of Clubs

Dealer's hand:
9 of Spades

Type h for new card, s to stop: h
You drew the 6 of Spades

Type h for new card, s to stop: s

Dealer drew the 3 of Spades
Dealer drew the 4 of Spades
Dealer drew the 8 of Hearts
Dealer went bust, you win!

The final scores were player: 20; dealer: 24
```

2. The (New) `card` Class

We'll be employing a slightly re-written version of the Card class from a prior lab.[3] You do not need to change anything in the `card` module, but you may find it useful to look at a non-trivial class definition employing class variables when you write your own class definitions in this lab or for course assignments.

3. Completing the Blackjack class definition

You should proceed in an **iterative** fashion: For **each** subsection that asks you to do some implementation,

(1) Read the directions in this handout and the specification of the relevant methods in the blackjack.py file.
(2) Look at the appropriate test cases in blackjack_checks.py, to better understand the usage and goals of the code you will write.
(3) Replace the "pass" lines and write the appropriate code.
(4) Check your code using `blackjack_checks.py`. We say "check" instead of "test" because we haven't supplied completely bulletproof suites of test cases, but still, you do not need to add test cases to it.

[3] The major changes: removed poker-specific functions, changed global variables to class variables. Why did we originally have poker functions in the module `card`? At the time of that lab, it was convenient to put all the required code in one place so that students wouldn't have too many files to deal with. But now we've gotten to the point where exemplifying better file organization is appropriate.

Make sure you've checked your work for a given subsection *before* moving on to the next one. This is important because many of the methods here build on earlier ones.

3.1. **Implement and test `__init__`.** Implement `__init__` so that it initializes the three instance attributes of `Blackjack`. For this part, you will probably want to make use of standard list operations. For reference, look at section 5.1 in the Python library at

`http://docs.python.org/2/tutorial/datastructures.html`

Our solution is three lines long. Write yours here:

3.2. **Read over but don't change helper function `_score`.** Note the leading underscore in this function's name, indicating that it is meant to be a private helper function.

Also note that it is a function that is not a method. Given the specification of `_score`, why does it make sense that this function is not a method?

3.3. **Implement `dealerScore()` and `playerScore()`.** Make use of function `_score`. Write your code for `dealerScore` here:

3.4. **Implement and test `playerBust()` and `dealerBust()`.** Your implementation should use `dealerScore()` and `playerScore()` as helper methods. Write your code for `playerBust()` here:

3.5. **Implement and test `__str__`.** . Note that this method is "higher up" in the file, just after `__init__`, as is conventional. Use `dealerScore()` and `playerScore()` as helper methods.

3.6. **Play some Blackjack!** This last part is just for fun. Run `blackjack.py` as a script:

```
python blackjack.py
```

Follow the directions on the screen. The command 'h' is for 'hit', and 's' is for stay.

Our dealer is following a common protocol: they continue to hit while their hand is under 17, but once their hand reaches 17 or more, they stay (if they haven't already gone bust). See if you can use this to your advantage.

3.6.1. *Optional reading: the code for playing a game.* You might find it interesting to look over the code in blackjack.py for playing a game. It starts at the line `def play_a_game():`. There is some new content: the use of while-loops, the command `raw_input`, and the use of a negative list index; but we've endeavored to make this code as readable for you as possible.

4. Facilitating checking-off

Here's a checklist to be ready to quickly demonstrate your work to a staff member.

- ☐ You have a copy of this handout with all the white boxes filled in.
- ☐ You are ready to show that your code passes the given test cases by running `python blackjack_checks.py` in the command shell.
- ☐ You are ready to show that you can play the blackjack game by running `python blackjack.py` in the command shell.