CSCE 110: Programming I

Lab #6 (100 points)

Due: Sunday, October 9th by 11:59pm

1 Please make sure you understand the following.

For this assignment, you are only allowed to use what we have discussed during the first 6 weeks of class. If we haven't discussed it, you cannot use it in your program. Your programs must be composed of user-defined functions. If you fail to follow the above instructions, you will receive a 0 for your programs.

Please label your Python programs `q<num>.py`, where `<num>` is the question number. Take your time and make sure you understand everything in this lab before getting started. Also, make sure your programs match the output EXACTLY as given for each question.

2 Lab Questions

1. *Primary arithmetic.* Write a program (called q1.py) that adds two numbers and counts the number of carry operations. You will define your own user-defined functions for this problem. Your solution must make use of at least two user-defined functions to receive full credit.

   Programming tips. There are a number of different ways to tackle this problem. Consider first working on printing out the addition equation as shown in the following examples. Then, work on computing the number of carries required for the addition. Finally, a built-in function that you may find useful to incorporate in your program is `max()`. For example, to find the maximum of the 3 numbers 54, 198, and -34, use `max(54, 198, -34)`, which returns 198.

Example #1. After each prompt for a number, the user enters 458 and 141, respectively (lines 1–2). The program prints the addition equation with the sum of the two numbers (lines 4–8). Afterwards, the program shows that there are no carry operations (line 10).

```
 1  Number #1: 458
 2  Number #2: 141
 3
 4    458
 5  +
 6    141
 7  -----
 8    599
 9
10  Carries: 0
```

Example #2. Follows similarly to Example #1. Here, one carry operation is required.

```
 1  Number #1: 9
 2  Number #2: 8
 3
 4     9
 5  +
 6     8
 7  ----
 8    17
 9
10  Carries: 1
```

Example #3. The numbers (689991 and 9839) entered by the user are not the same length (lines 1–2). The input is then aligned with the resulting sum (lines 4–8). There are 4 carry operations (line 10).

```
 1  Number #1: 689991
 2  Number #2: 9839
 3
 4    689991
 5  +
 6      9839
 7  --------
 8    699830
 9
10  Carries: 4
```

Example #4. Every addition of the digits results in a carry operation. There are 7 carries (line 10).

```
 1  Number #1: 981
 2  Number #2: 9999999
 3
 4        981
 5  +
 6     9999999
 7  ----------
 8    10000980
 9
10  Carries: 7
```

2. *Blackjack Dice.* Write a program (called q2.py) to simulate the game of Blackjack Dice. Blackjack Dice is based on two 11-sided dice, where each die represents the numbers from 1 to 11. The objective of this game is to roll 21. If you go over, it's considered a bust and you lose. Your score (without going over 21) must beat the dealer or you lose.

   - *Rules for the Player.* The player begins by rolling the dice and the program adds their corresponding numbers. After the initial roll of the two dice, a player may choose to roll a single die as many times as they wish, adding the resulting number from each roll to their total. If the total exceeds 21, the dealer wins. If the player gets 21 after the first roll, this is considered Blackjack and leads to an automatic win.
   - *Rules for the Dealer.* The dealer begins by rolling two dice and adds the numbers to get a total. The dealer continues rolling the dice until it reaches a total greater than 16. If the dealer's score (21 or less) is greater than the player's score, the dealer wins. The player wins with a total (21 or less) that is greater than the dealer's total or when the dealer busts (goes over 21). The dealer wins all ties.

   Your program will consist of 4 functions: `roll_die()`, `player()`, `dealer()`, and `main()`, which rolls an 11-sided die, implements what happens on the player's (or user's) turn, implements what happens on the dealer's turn, and drives the program, respectively.

   Attached to this assignment is q2.py, which provides a skeleton of the code for this program. Your goal is to complete the code for the program using the template provided. The roll_die() function has been completed for you. **You must complete the code for the remaining functions (`player()`, `dealer()`, and `main()`) as described by the comments in `q2.py`.**

Tips for Writing Your Program.

a) Study the example output below to get a feel for what your program is supposed to do. To keep everything from scrolling to quickly during the dealer's turn, the user will press the enter key to unveil slowly what happens after each roll.

b) Once you understand what to do, work on the implementation for the player. Your work will go inside the `player()` function.

c) Write the code for the dealer, which goes inside the `dealer()` function.

d) Write the code to determine the winner, which will go inside the `main()` function.

e) To debug your program, make use of print statements. For example, print the result of randomly rolling the dice so that you can check the logic of your program.

Example #1. The user input is on lines 4, 8 12, 17, and 21. That is, when prompted the user enters "s" or "r" to determine whether to stay or roll. When it's the dealer's turn, the user simply presses the enter key when prompted.

The user goes first (line 1). The first roll is a 4 and 2 for a total of 6 (line 2 and 3). The user decides to roll again (line 4). 5 is rolled for a total of 11 (lines 6 and 7). The user chooses to roll again (line 8). 8 is rolled (line 10). With a total of 19, the user chooses to stay (lines 11 and 12). Play continues with the dealer (line 14). The dealer rolls a 2 and 8 on the first roll for a total of 10 (lines 15 and 16). The user is prompted to press the enter key to continue (line 17). The dealer then rolls a 1 for a total of 11 (lines 19 and 20). The user presses the enter again (line 21). The dealer rolls a 6 for a total of 17 (lines 23 and 24). The dealer must stop rolling once a total of 17 or more has been reached. Since the user's total is greater than the dealer's total, the game is over (line 26) and the user wins (line 27).

```
 1  ************ YOUR TURN ************
 2  Roll: 4 2
 3  Total: 6
 4  (s)tay or (r)oll? r
 5
 6  Roll: 5
 7  Total: 11
 8  (s)tay or (r)oll? r
 9
10  Roll: 8
11  Total: 19
12  (s)tay or (r)oll? s
13
14  ********** DEALER'S TURN **********
15  Roll: 2 8
16  Total: 10
17  Press <enter> to continue ...
18
19  Roll: 1
20  Total: 11
21  Press <enter> to continue ...
22
23  Roll: 6
24  Total: 17
25
```

```
************ GAME OVER ************
You win!
```

Example #2. The user wins by getting Blackjack on the first roll. Since the user has Blackjack, the dealer doesn't roll the dice.

```
************ YOUR TURN ************
Roll: 10 11
Total: 21

Blackjack!

************ GAME OVER ************
You win!
```

Example #3. The user wins since the dealer busts with a total over 21.

```
************ YOUR TURN ************
Roll: 4 6
Total: 10
(s)tay or (r)oll? r

Roll: 5
Total: 15
(s)tay or (r)oll? r

Roll: 2
Total: 17
(s)tay or (r)oll? s

********** DEALER'S TURN **********
Roll: 11 4
Total: 15
Press <enter> to continue ...

Roll: 1
Total: 16
Press <enter> to continue ...

Roll: 7
Total: 23

Bust!

************ GAME OVER ************
You win!
```

Example #4. The dealer wins since the player has a total over 21. Since the user busts, the dealer to roll the dice.

```
************ YOUR TURN ************
Roll: 10 4
Total: 14
(s)tay or (r)oll? r

Roll: 11
Total: 25

Bust!

********** GAME OVER **********
Dealer wins!
```

Example #5. Both the user and dealer have a total of 21. Since there's a tie, the dealer wins.

```
************ YOUR TURN ************
Roll: 1 11
Total: 12
(s)tay or (r)oll? r

Roll: 9
Total: 21

********** DEALER'S TURN **********
Roll: 2 5
Total: 7
Press <enter> to continue ...

Roll: 4
Total: 11
Press <enter> to continue ...

Roll: 1
Total: 12
Press <enter> to continue ...

Roll: 9
Total: 21

************ GAME OVER ************
Dealer wins!
```