Programming Project 04

This assignment is worth 40 points (4.0% of the course grade) and must be **completed and turned in before 11:59 PM on Monday, October 8, 2018.**

Assignment Overview
(learning objectives)
This assignment will give you more experience on the use of:

1. Functions
2. iteration
3. string

The goal of this project is to play a simplified game of craps.

Assignment Background

A popular dice game is Craps. In this project you will handle the rolling of dice and wagers while a user plays the game. Wagering for craps is complicated; we use a simplified version here.

The rules: A player rolls two dice. Each die has six faces. These faces contain 1, 2, 3, 4, 5, and 6 spots. After the dice have come to rest, the sum of the spots on the two upward faces is calculated. If the sum is 7 or 11 on the first throw, the player wins (this is called a "Natural win"). If the sum is 2, 3, or 12 on the first throw (called "craps"), the player loses (i.e. the "house" wins). If the sum is 4, 5, 6, 8, 9, or 10 on the first throw, then the sum becomes the player's "point." To win, you must continue rolling the dice until you "make your point." The player loses by rolling a 7 before making the point.

Project Description

Your program must meet the following specifications. Note that control gets complex to play this game: I had 3 while loops and almost 20 if statements.

1. Your program must prompt for an initial player's bank balance (from which wagers will be subtracted or added). Note that all dollar amounts in this game will be ints.
2. Before each game (i.e. before the first roll of the game) prompt the user for a wager. Wagers must be less than or equal to the bank balance. If not, keep asking for a wager until a valid one is entered. (Hint: use a while loop).
3. One game is played with a sequence of two dice rolls. A game can end after one roll of the two dice (e.g. a 7 was rolled) or it can continue with an unlimited number of rolls (Hint: use a while loop). Note that the rules are different on the first roll of the game than on subsequent rolls. (Hint: use a Boolean to indicate that it is the first roll, e.g. I used one named "first_roll" that was initially True and after the first roll was changed to False and not changed for the subsequent rolls of the game.)
4. After you play each game update the player's bank balance by adding or subtracting the wager depending on whether they won or lost.
5. Prompt the user to see if they want to play another game. If "yes", play another game. (Hint: use a while loop).
6. If the user is to play another game, prompt the user to see if they wish to add to their balance. If "yes", then prompt for a value and update the player's bank balance.
7. You have to use the ten functions in the provided proj04.py skeleton. You have to implement and use the

a. `display_game_rules()`: We provide this function.
b. `get_bank_balance()`: Prompt the player for an initial bank balance (from which wagering will be added or subtracted). The player-entered bank balance is returned as an int.
c. `add_to_bank_balance(balance)`: Prompts the player for an amount to add to the balance. The balance is returned as an int.
d. `get_wager_amount()`: Prompts the player for a wager on a particular roll. The wager is returned as an int.
e. `is_valid_wager_amount(wager, balance)`: Checks that the wager is less than or equal to the balance; returns True if it is; False otherwise.
f. `roll_die()`: Rolls one die. This function should randomly generate a value between 1 and 6 inclusively and returns that value as an int. Use the `randint()` function called as `randint(1,6)` for this project. We provide the import at the top of the file. (For Mimir testing the default is to use the cse231_random library which isn't actually random but generates specific test cases. To actually play the game comment out the cse231 version and use the actual Python random.)
g. `calculate_sum_dice(die1_value, die2_value)`: Sums the values of the two die and returns the sum as an int.
h. `first_roll_result(sum_dice)`: The function determines the result on the first roll of the pair of dice. A string is returned. You are required to use at least one Boolean operator "or" or "and" in this function (I used 4).
    i. If the sum is 7 or 11 on the roll, the player wins and "win" is returned.
    ii. If the sum is 2, 3, or 12 on the first throw (called "craps"), the player loses (i.e. the "house" wins) and "loss" is returned.
    iii. If the sum is 4, 5, 6, 8, 9, or 10 on the first throw, then the sum becomes the player's "point" and "point" is returned.
i. `subsequent_roll_result(sum_dice, point_value)`: The function determines the result on the subsequent rolls of the pair of dice. A string is returned.
    i. If `sum_dice` is the `point_value`, then "point" is returned
    ii. If `sum_dice` is 7, then "loss" is returned
    iii. Otherwise, "neither" is returned.

j. `main()`: Takes no input. Returns nothing. Call the functions from here. That, the game is played in this function.

Assignment Deliverable

The deliverable for this assignment is the following file:

`proj04.py` – the source code for your Python program

Be sure to use the specified file name and to submit it for grading via the **Mimir** before the project deadline.

Assignment Notes

1. To clarify the project specifications, sample output is appended to the end of this document.
2. Items 1-9 of the Coding Standard will be enforced for this project.
3. We provide a proj04.py program for you to start with.
4. You are not allowed to use advanced data structures such as list, dictionaries, classes….
5. If you "hard code" answers, you will receive a grade of zero for the whole project. An example

of hard coding is to simply print an average rather than calculating an average and then printing the calculated average.

Test Cases

Test 1

```
A player rolls two dice. Each die has six faces.
          These faces contain 1, 2, 3, 4, 5, and 6 spots.
          After the dice have come to rest,
          the sum of the spots on the two upward faces is calculated.
          If the sum is 7 or 11 on the first throw, the player wins.
          If the sum is 2, 3, or 12 on the first throw (called "craps"),
          the player loses (i.e. the "house" wins).
          If the sum is 4, 5, 6, 8, 9, or 10 on the first throw,
          then the sum becomes the player's "point."
          To win, you must continue rolling the dice until you "make your point."
          The player loses by rolling a 7 before making the point.
Enter an initial bank balance (dollars): 50
Enter a wager (dollars): 10
Die 1: 3
Die 2: 4
Dice sum: 7
Natural winner.
You WIN!
Balance: 60
Do you want to continue? yes
Do you want to add to your balance? no
Enter a wager (dollars): 10
Die 1: 1
Die 2: 2
Dice sum: 3
Craps.
You lose.
Balance: 50
Do you want to continue? yes
Do you want to add to your balance? no
Enter a wager (dollars): 10
Die 1: 5
Die 2: 6
Dice sum: 11
Natural winner.
You WIN!
Balance: 60
Do you want to continue? yes
Do you want to add to your balance? no
Enter a wager (dollars): 10
Die 1: 2
Die 2: 3
Dice sum: 5
*** Point: 5
Die 1: 2
Die 2: 2
Dice sum: 4
Die 1: 3
Die 2: 3
Dice sum: 6
Die 1: 1
Die 2: 4
Dice sum: 5
```

```
You matched your Point.
You WIN!
Balance: 70
Do you want to continue? yes
Do you want to add to your balance? no
Enter a wager (dollars): 10
Die 1: 2
Die 2: 4
Dice sum: 6
*** Point: 6
Die 1: 2
Die 2: 3
Dice sum: 5
Die 1: 4
Die 2: 5
Dice sum: 9
Die 1: 3
Die 2: 4
Dice sum: 7
You lose.
Balance: 60
Do you want to continue? no
Game is over.
```

Test 2

```
A player rolls two dice. Each die has six faces.
          These faces contain 1, 2, 3, 4, 5, and 6 spots.
          After the dice have come to rest,
          the sum of the spots on the two upward faces is calculated.
          If the sum is 7 or 11 on the first throw, the player wins.
          If the sum is 2, 3, or 12 on the first throw (called "craps"),
          the player loses (i.e. the "house" wins).
          If the sum is 4, 5, 6, 8, 9, or 10 on the first throw,
          then the sum becomes the player's "point."
          To win, you must continue rolling the dice until you "make your point."
          The player loses by rolling a 7 before making the point.
Enter an initial bank balance (dollars): 10
Enter a wager (dollars): 20
Error: wager > balance. Try again.
Enter a wager (dollars): 10
Die 1: 3
Die 2: 4
Dice sum: 7
Natural winner.
You WIN!
Balance: 20
Do you want to continue? yes
Do you want to add to your balance? yes
Enter how many dollars to add to your balance: 10
Balance: 30
Enter a wager (dollars): 10
Die 1: 1
Die 2: 2
Dice sum: 3
Craps.
You lose.
Balance: 20
Do you want to continue? yes
Do you want to add to your balance? no
Enter a wager (dollars): 20
```

```
Die 1: 5
Die 2: 6
Dice sum: 11
Natural winner.
You WIN!
Balance: 40
Do you want to continue? yes
Do you want to add to your balance? no
Enter a wager (dollars): 10
Die 1: 2
Die 2: 3
Dice sum: 5
*** Point: 5
Die 1: 2
Die 2: 2
Dice sum: 4
Die 1: 3
Die 2: 3
Dice sum: 6
Die 1: 1
Die 2: 4
Dice sum: 5
You matched your Point.
You WIN!
Balance: 50
Do you want to continue? yes
Do you want to add to your balance? no
Enter a wager (dollars): 50
Die 1: 2
Die 2: 4
Dice sum: 6
*** Point: 6
Die 1: 2
Die 2: 3
Dice sum: 5
Die 1: 4
Die 2: 5
Dice sum: 9
Die 1: 3
Die 2: 4
Dice sum: 7
You lose.
Balance: 0
Do you want to continue? yes
Do you want to add to your balance? no
You don't have sufficient balance to continue.
Game is over.
```

Grading Rubric

```
Computer Project #04                                Scoring Summary
General Requirements:
  ( 4 pts) Coding Standard 1-9
             (descriptive comments, function headers, etc...)

Implementation:
  ( 1 pts) get_bank_balance function (no Mimir test)
  ( 1 pts) add_to_bank_balanc function (no Mimir test)
  ( 1 pts) get_wager_amount function (no Mimir test)
```

(2 pts) roll_die function
(2 pts) is_valid_wager_amount function
(2 pts) calculate_sum_dice function
(6 pts) first_roll_result function
(6 pts) subsequent_roll_result function
(8 pts) Test1
(7 pts) Test2

Note: hard coding an answer earns zero points for the whole project
-10 points for not using main()