CSCE 110: Programming I

Lab #3 (100 points)

Due: Sunday, September 18 by 11:59pm

Directions: Please label your Python programs `q<num>.py`, where `num` is the question number. For example, your solution to the first question, will be stored in the file `q1.py`.

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. This is important as one of the keys to being a good programmer is attention to details.

1 Please make sure you understand the following.

For this assignment, you are only allowed to use what we have discussed in the first 3 weeks of class. If we haven't discussed it in class, you cannot use it in your program. Have fun!

2 Lab Questions

1. *Counting change.* Write a Python program (called q1.py) that for a given change amount (in dollars and cents), reports the maximum number of dollars, quarters, dimes, nickels, and pennies. *Note:* You will find it easier to convert the change amount to cents (an integer) to make change.

   Example #1. At the prompt, the user enters 11.42 (line 1). The program outputs the number of dollars, quarters, dimes, nickels, and pennies to make $11.42 (lines 3–7).

```
1 Enter the change amount: 11.42
2
3 11 dollars
4 1 quarters
5 1 dimes
6 1 nickels
7 2 pennies
```

Example #2. At the prompt, the user enters 4.32 (line 1). The program then outputs the maximum amount for each denomination (lines 3–7).

```
1 Enter the change amount: 4.32
2
3 4 dollars
4 1 quarters
5 0 dimes
6 1 nickels
7 2 pennies
```

Programming tips. Because of the way Python handles multiplying floats, you'll have to make an additional adjustment. For example, suppose the change amount is 0.29. If we simply multiply by 100, the answer is incorrect (lines 1 and 3 in the Python shell example below). The result should be 29 cents.

```
1 >>> 0.29 * 100
2 28.999999999999996
3 >>> int(0.29 * 100)
4 28
5 >>> 0.29 * 100 + 0.5
6 29.499999999999996
7 >>> int(0.29 * 100 + 0.5)
8 29
9 >>> 0.15 * 100 + 0.5
10 15.5
11 >>> int(0.15 * 100 + 0.5)
12 15
```

You will need to add 0.5 to your expression to convert the change to cents correctly (see lines 5 and 7). Line 11 shows how the expression works for converting the change amount of 0.15 to 15 cents. So, when converting the user's change to cents, make sure to add 0.5 before converting to an integer.

Example #3. Make sure to read the programming tips above and modify your program accordingly. Otherwise, your program will not produce the right output for 29 cents.

```
1 Enter the change amount: 0.29
2
3 0 dollars
4 1 quarters
5 0 dimes
6 0 nickels
7 4 pennies
```

2. *Counting change, part 2.* Let's revisit Question #2. Write a Python program (called q2.py) so that if a denomination has a value of 0, it's not printed.

   Example #1. When returning $4.32 in change, there are no dimes.

```
Enter the change amount: 4.32

4 dollars
1 quarters
1 nickels
2 pennies
```

   Example #2. For $11.00, there are no quarters, dimes, nickels or pennies.

```
Enter the change amount: 11

11 dollars
```

   Example #3. All denominations are returned for $9.43.

```
Enter the change amount: 9.43

9 dollars
1 quarters
1 dimes
1 nickels
3 pennies
```

3. *Cramer's rule.* Write a Python program (called q3.py) to compute Cramer's rule to solve 2 x 2 linear equations. 2 x 2 linear equations have the following form.

$$ax + by = e$$
$$cx + dy = f$$

   Given the known values (a, b, c, d, e, and f), the goal is to solve for x and y. Both x and y can be computed by the following equations.

$$x = \frac{ed-bf}{ad-bc}, \quad y = \frac{af-ec}{ad-bc}$$

   A solution to Cramer's rule is not found when the denominator of the equations for x or y is 0. When this happens, print **No unique solution found!**

   Programming tips. Since you can't divide by 0, make sure to check that the denominator is not 0 before doing the division. If the denominator is 0, there is no unique solution.

Example #1. The user enters the values for the knowns a, b, c, d, e, and f (lines 1–6). Then, the two equations to solve are printed (lines 8–10). The program then outputs the solutions for x and y (lines 12–13).

```
1  Enter a: 9.0
2  Enter b: 4
3  Enter c: 3
4  Enter d: -5
5  Enter e: -6
6  Enter f: -21.0
7
8  Solving for the following two equations.
9  9.0x + 4.0y = -6.0
10 3.0x + -5.0y = -21.0
11
12 x =  -2.0
13 y =  3.0
```

Example #2. Similar to previous example, but there is no unique solution since $ad - bc = 0$.

```
1  Enter a: 10
2  Enter b: 10
3  Enter c: 1
4  Enter d: 1
5  Enter e: 1
6  Enter f: -10
7
8  Solving for the following two equations.
9  10.0x + 10.0y = 1.0
10 1.0x + 1.0y = -10.0
11
12 No unique solution found!
```

Example #3. Another interesting example.

```
1  Enter a: -0.321
2  Enter b: 1.9983
3  Enter c: 876.43
4  Enter d: 10
5  Enter e: -97
6  Enter f: 10001
7
8  Solving for the following two equations.
9  -0.321x + 1.9983y = -97.0
10 876.43x + 10.0y = 10001.0
11
12 x =  11.9430276624
13 y =  -46.6227734176
```

4. *Guessing Birthdays.* Write a program (called q4.py) that guesses birthdays. You can determine the date of the month when someone was born by asking five questions. Each question asks whether the day is in each of the five sets of numbers, which are shown below in Figure 1.

1	3	5	7
9	11	13	15
17	19	21	23
25	27	29	31

Set 1

2	3	6	7
10	11	14	15
18	19	22	23
26	27	30	31

Set 2

4	5	6	7
12	13	14	15
20	21	22	23
28	29	30	31

Set 3

8	9	10	11
12	13	14	15
24	25	26	27
28	29	30	31

Set 4

16	17	18	19
20	21	22	23
24	25	26	27
28	29	30	31

Set 5

Figure 1: The five sets of numbers for guessing birthdays.

The birthday is the sum of the first numbers in the sets where the date appears. For example, if the birthday is 19, it appears in Set 1, Set 2, and Set 5. The first numbers in these three sets are 1, 2, and 16. Their sum is 19.

Example #1. First, ask the user if their birthday is in Set 1 (line 1). The program then outputs the numbers in Set 1 (lines 3–6). The user answers the question by entering either 'n' for no or 'y' for yes (line 8). Next, the second question is asked (line 10) followed by outputting the numbers in Set 2 (lines 12–15). The user answers by entering 'n' or 'y' (line 17). The program continues executing in the above manner for the remaining questions. Once the five questions have been asked, the program outputs the birthday (line 46).

```
1  Question #1: Is your birthday in Set 1?
2
3            1  3  5  7
4            9 11 13 15
5           17 19 21 23
6           25 27 29 31
7
8  Enter (n)o or (y)es: y
9
10 Question #2: Is your birthday in Set 2?
11
12           2  3  6  7
13          10 11 14 15
14          18 19 22 23
```

```
         26 27 30 31

Enter (n)o or (y)es: y

Question #3: Is your birthday in Set 3?

          4  5  6  7
         12 13 14 15
         20 21 22 23
         28 29 30 31

Enter (n)o or (y)es: n

Question #4: Is your birthday in Set 4?

          8  9  10 11
         12 13 14 15
         24 25 26 27
         28 29 30 31

Enter (n)o or (y)es: n

Question #5: Is your birthday in Set 5?

          16 17 18 19
         20 21 22 23
         24 25 26 27
         28 29 30 31

Enter (n)o or (y)es: y

Your birthday is 19!
```

Example #2. Another interesting example.

```
Yoution #1: Is your birthday in Set 1?

          1  3  5  7
          9 11 13 15
         17 19 21 23
         25 27 29 31

Enter (n)o or (y)es: n

Question #2: Is your birthday in Set 2?

          2  3  6  7
         10 11 14 15
```

```
         18 19 22 23
         26 27 30 31

Enter (n)o or (y)es: y

Question #3: Is your birthday in Set 3?

          4  5  6  7
         12 13 14 15
         20 21 22 23
         28 29 30 31

Enter (n)o or (y)es: n

Question #4: Is your birthday in Set 4?

          8  9  10 11
         12 13 14 15
         24 25 26 27
         28 29 30 31

Enter (n)o or (y)es: y

Question #5: Is your birthday in Set 5?

          16 17 18 19
         20 21 22 23
         24 25 26 27
         28 29 30 31

Enter (n)o or (y)es: y

Your birthday is 26!
```