CSC 401 ASSIGNMENT THREE

Due Date: Tuesday, Oct. 1st by 11:58 PM

The purpose of this assignment is to assess your understanding of

- User defined functions and parameter passing (Section 3.3 and 3.5)
- List methods (Section 2.3 pages 31-33)
- String formatting (Section 4.1 and 4.2)
- Iteration Structure – for loop (Section 3.2)
- Reading files (Section 4.3)

SUBMISSION

- **Include your full name as a comment in the first line of your Python program**
- **Include the problem number as a comment in the second line of our Python program**
- Save each program to a separate file labeled as YourName_hw3_1.py, YourName_hw3_2.py, YourName_hw3_3.py accordingly
- Upload each file to Submissions folder in D2L.

PROBLEMS

Note: you may not use Python statements, functions, data types, etc. that where not discussed in the reading assignment or the lecture notes/videos for week 3 or previous weeks. This is a class for students who have not programmed before and I expect everyone to code on the same level. If you have a better way of writing the code, then upload two versions: one that codes according to the specifications and the other that demonstrates advanced programming techniques.

I encourage you to use computational thinking to solve the problems. These are straight-forward solutions, but developing a good habit of analyzing the problem and describing the steps will serve you well as the problems get more complex.

I encourage you to test your code thoroughly, but I am **not** requiring that you include you test cases as part of your programs. **I do require that you code according to the problem specifications.**

PROBLEM ONE

Write a Python program that separates a list of up to 100 positive and negative integers into a sorted list of even integers and a sorted list of odd integers. There are two user-defined functions in this program.

PART ONE (10 POINT)
USER-DEFINED FUNCTION, LIST, ITERATION (FOR LOOP), DECISION

Write the function **evenOdd (num)** that has 1 argument, **num**, a positive integer no greater than 100. This function returns a list of sorted even integers and a list of sorted odd integers.

- Create a list, **lst**, of positive and negative integers using the following code:

```
import random   # include outside of function
for i in range(num):
    lst.append(random.randint(-50, 50))
```

  lst will contain **num** positive and negative integers between -50 and 50. There may be duplicate numbers.
- Divide **lst** into 2 lists; One that contains all the even integers and one that contains all the odd integers.
  - Even integers are divisible by 2; odd integers are not divisible by 2.
- The function returns the two lists. Each list is sorted in ascending order. (use list sort method)

PART TWO (10 POINTS)
USER-DEFINED FUNCTION, LIST, ITERATION (FOR LOOP), DECISION, FORMATTING

Write the function main( ) that has no arguments.

- main() has no arguments
  - Ask the user to enter the number of integers they want in the list. The number may not be greater than 100.
  - Call the evenOdd() function passing the number the user entered to the function
  - Evaluate the returned results and displays the list in two formatted columns; one column of even numbers and another column of odd numbers
- You must pass arguments from the calling program function to the called program function. (you may **not** use global variables i.e. variables defined outside of either function.)

Sample cases:

```
>>> main()
Enter number of integers in list: 20
even    odd
-46     -45
-40     -39
-32     -13
-28     -13
-22      -7
 -4      -5
  0      17
  2      25
 20
 34
 40
 48
>>> main()
Enter number of integers in list: 3
even    odd
-46      11
         23
>>> main()
Enter number of integers in list: 3
even    odd
-38       3
 36
>>> main()
Enter number of integers in list: 3
even    odd
        -29
         -7
          7
```

PROBLEM 2 (10 POINTS)
ITERATION (FOR LOOP), FORMATTING

The monthly payments for a given loan are divided into amounts that apply to the principal and to the interest. For example: if you make a monthly payment of $500, only a portion of the $500 goes to the principal and the remainder is the interest payment.

- monthly interest = monthly interest rate x balance of principal

- balance of principal = monthly payment – monthly interest
- monthly payment =
  (monthly interest rate x balance of principal) / (1 – (1 + monthly interest rate) $^{-months}$)

Write a Python program that accepts input from a user for a loan amount (principal), annual interest rate (convert to monthly interest rate), and loan period (in number of year, convert to number of months) and displays a table with six columns: payment number, monthly payment, interest, principal, remaining balance after the payment, and the total interest paid to name.
For example: a loan payment schedule for a 1-year (12 months) loan of $5000 with a 12% annual interest rate (.01 monthly interest rate) is:

```
length of loan period in years: 2
loan amount: 10000
annual interest rate as percent:12
No.     Payment   Interest  Principal    Balance Total Interest
 1      470.73     100.00     370.73    9629.27     100.00
 2      470.73      96.29     374.44    9254.82     196.29
 3      470.73      92.55     378.19    8876.64     288.84
 4      470.73      88.77     381.97    8494.67     377.61
 5      470.73      84.95     385.79    8108.88     462.55
 6      470.73      81.09     389.65    7719.23     543.64
 7      470.73      77.19     393.54    7325.69     620.84
 8      470.73      73.26     397.48    6928.21     694.09
 9      470.73      69.28     401.45    6526.76     763.37
10      470.73      65.27     405.47    6121.29     828.64
11      470.73      61.21     409.52    5711.77     889.85
12      470.73      57.12     413.62    5298.16     946.97
13      470.73      52.98     417.75    4880.40     999.95
14      470.73      48.80     421.93    4458.47    1048.76
15      470.73      44.58     426.15    4032.32    1093.34
16      470.73      40.32     430.41    3601.91    1133.67
17      470.73      36.02     434.72    3167.19    1169.69
18      470.73      31.67     439.06    2728.13    1201.36
19      470.73      27.28     443.45    2284.68    1228.64
20      470.73      22.85     447.89    1836.79    1251.49
21      470.73      18.37     452.37    1384.42    1269.85
22      470.73      13.84     456.89     927.53    1283.70
23      470.73       9.28     461.46     466.07    1292.97
24      470.73       4.66     466.07       0.00    1297.63
```

Test your code with the sample data then test again with the loan period of half a year, i.e. .5 years.

PROBLEM THREE (10 POINTS)
READ FILE, LIST, DECISION

Write a function wordGame() that reads the text file, Pride_and_Prejudice.txt. (Be sure to save Pride_and_Prejudice.txt in the same folder in which you are saving your program.) Include code in this function that retrieves 2 random words from the file. Ask the user to guess which of the words the author used more often in the text. If the user picks the word with the higher count, then they have guessed correctly.
To pick a random word, import random and use the random.choice () method. To determine how often a word occurs (use the count method). Use a logical condition to determine if the user guessed correctly. As shown in the sample cases, be sure to include the chosen words in double quotes.
Note: There is only one function in this program. To run the function, type wordGame() at the IDLE prompt.

Sample cases :

```
>>> wordGame()
Which word did the writer use more often "such" or "liberty"? such
you are correct
>>> wordGame()
Which word did the writer use more often "their" or "would"? would
you are correct
>>> wordGame()
Which word did the writer use more often "the" or "her"? her
you are incorrect
```

Assignment Three Grading Rubric
Learning outcomes:

- Create user-defined functions
- Utilize string methods in text processing
- Read text file from within a Python program
- Use the String method format() to improve the readability of the output string
- Use iteration structure to repeat code
- Use methods from random library

Problem Numbers	Proficient 10 – 9	Nearing Proficiency 8 – 7	Needs Improvement 6 – 0
	Shows a comprehensive understanding of user-defined functions and parameter passing; string	Shows an adequate understanding of user-defined functions and parameter passing, string	Shows a minimal or no understanding of user-defined functions and parameter passing, string

	formatting, list methods, reading a file and iteration (for loops)	formatting, list methods and iteration (for loops)	formatting, list methods and iteration (for loops).
1	Correctly defines both functions as described EvenOdd() Accepts integer as an argument EvenOdd() Returns two lists in one return statement Output is in the format shown in the sample case	Adequately defines both functions as described EvenOdd() Accepts more than one argument EvenOdd() Returns only one list Output differs somewhat from that shown in the sample case	Functions have minimal or no correct code EvenOdd() does not accept any arguments EvenOdd() Does not return any lists Output missing data or not formatted as shown in the sample case.
2	Prompts user for all described data All calculations are correct Output is displayed in described format and as shown in example	User is prompted for only some of the described data Some calculations are correct Some of the output is not displayed in described format and as shown in the example	User is not prompted for any data All calculations are incorrect Output does not follow the described format or the example
3	Function correctly opens, read and closes the file Function correctly retrieves two random words and displays the words enclosed in quotes as shown in the example Function correctly compares word count and displays appropriate message as shown in examples	Function correctly opens files, uses inappropriate read method and closes file Function correctly retrieves the words and does not display them as shown in the example Function does not compare all the possible word count conditions	Function is not defined correctly File is not correctly read and/or is not closed Words are not randomly chosen as described Word count is incorrect Comparisons are not correctly coded Messages are incomplete or incorrect

For all problems	Program (functions) do not use any data types, statement, methods or operators that have not been presented in week 3 or prior week's lectures or reading assignments. Complete thorough testing Program (functions) have no syntax errors Programs (functions) execute with no run time errors All problem specifications are correctly coded	Program (functions) uses a data type, statement, method or operator that have not been presented in week 3 or prior week's lectures or reading assignments. Only tested with given data or partially tested Program (functions) have one or two syntax errors. Program (functions) do not execute because of a run time error Some deviations from specifications	Program (functions) uses more than one data type, statement, method or operator that have not been presented in week 3 or prior week's lectures or reading assignments. Minimal or no testing Programs (functions) have more than two syntax errors Program (functions) do not execute because of run-time errors Hardly follows the specifications