Textbook Assignment 2

Maximum earnable: 120 pt.
Due: 11:59 PM November 10, 2023

- Read the assignment carefully. *You will need to **write and execute several Python scripts**; and **submit your code work together with their results***.
- For this assignment, you are required to **submit a report** and ***Python* code** *using a provided template* (in the *.py* format).
  - Make sure your code produces the same results as the solutions included in your report; **otherwise, you will get penalties.**
- You are **allowed to re-use any of the code snippets from the lecture slides** while developing solutions to the problems.
- This is an individual work;
  - Submitting assignments or program codes written by others or acquired from the internet without explicit approval of the professor is regarded as cheating.
  - Showing or lending one's own homework to other student is also considered cheating that disturbs fair evaluation and hinders the academic achievement of the other student.
  - It is regarded as cheating if two or more students conduct their homework together and submit it individually when the homework is not a group assignment.
- When finished, submit your work to *Google Classroom.*

1. (10 pt.) Write a function month() that takes a number between 1 and 12 as input and returns the three-character abbreviation of the corresponding month. Do this without using an if statement, just string operations. *Hint:* Use a string to store the abbreviations in order.

Include your solutions and explain your code in the report. Also, implement the solution in Python using the supplied file (Q1.py). Make sure all the results are reproducible for grading.

```
>>> month(1)

'Jan'

>>> month(11)

'Nov'
```

2. (15 pt.) Write function vowelCount() that takes a string as input and counts and prints the number of occurrences of vowels in the string.

Include your solutions and explain your code in the report. Also, implement the solution in Python using the supplied file (Q2.py). Make sure all the results are reproducible for grading.

```
>>> vowelCount('Le Tour de France')
a, e, i, o, and u appear, respectively, 1, 3, 0, 1, 1 times.
```

3. (15 pt.) The cryptography function crypto() takes as input a string (i.e., the name of a file in the current directory). The function should print the file on the screen with this modification: Every occurrence of string 'secret' in the file should be replaced with string 'xxxxxx'.

Include your solutions and explain your code in the report. Also, implement the solution in Python using the supplied file (Q3.py). Make sure all the results are reproducible for grading.

```
>>> crypto('crypto.txt')
I will tell you my xxxxxx. But first, I have to explain why it is a xxxxxx.

And that is all I will tell you about my xxxxxx.
```

4. (15 pt.) Write a function stats() that takes one input argument: the name of a text file. The function should print, on the screen, the number of lines, words, and characters in the file; your function should open the file only once.

Include your solutions and explain your code in the report. Also, implement the solution in Python using the supplied file (Q4.py). Make sure all the results are reproducible for grading.

```
>>> stats('example.txt')

line count: 3
word count: 20
character count: 98
```

5. (15 pt.) Write function pay() that takes as input an hourly wage and the number of hours an employee worked in the last week. The function should compute and return the employee's pay. Overtime work should be paid in this way: Any hours beyond 40 but less than or equal 60 should be paid at 1.5 times the regular hourly wage. Any hours beyond 60 should be paid at 2 times the regular hourly wage.

Include your solutions and explain your code in the report. Also, implement the solution in Python using the supplied file (Q5.py). Make sure all the results are reproducible for grading.

```
>>> pay(10, 35)
350
>>> pay(10, 45)
475.0
>>> pay(10, 61)
720.0
```

6. (10 pt.) Implement function leap() that takes one input argument—a year—and returns True if the year is a leap year and False otherwise. (A year is a leap year if it is divisible by 4 but not by 100, unless it is divisible by 400 in which case it is a leap year. For example, 1700, 1800 and 1900 are not leap years but 1600 and 2000 are.)

Include your solutions and explain your code in the report. Also, implement the solution in Python using the supplied file (Q6.py). Make sure all the results are reproducible for grading.

```
>>> leap(2008)
True
>>> leap(1900)
False
>>> leap(2000)
True
```

7. (10 pt.) Implement function fib() that takes a nonnegative integer n as input and returns the nth Fibonacci number.

Include your solutions and explain your code in the report. Also, implement the solution in Python using the supplied file (Q7.py). Make sure all the results are reproducible for grading.

```
>>> fib(0)
1
>>> fib(4)
5
>>> fib(8)
34
```

8. (10 pt.) Implement a function mystery() that takes as input a positive integer n and answers this question: How many times can n be halved (using integer division) before reaching 1? This value should returned.

Include your solutions and explain your code in the report. Also, implement the solution in Python using the supplied file (Q8.py). Make sure all the results are reproducible for grading.

```
>>> mystery(4)
2
>>> mystery(11)
3
>>> mystery(25)
4
```

9. (20 pt.) An inversion in a sequence is a pair of entries that are out of order. For example, the characters F and D form an inversion in string 'ABBFHDL' because F appears before D; so do characters H and D. The total number of inversions in a sequence (i.e., the number of pairs that are out of order) is a measure of how *unsorted* the sequence is. The total number of inversions in 'ABBFHDL' is 2. Implement function inversions() that takes a sequence (i.e., a string) of uppercase characters A through Z and returns the number of inversions in the sequence.

Include your solutions and explain your code in the report. Also, implement the solution in Python using the supplied file (Q9.py). Make sure all the results are reproducible for grading.

```
>>> inversions('ABBFHDL')
2
>>> inversions('ABCD')
0
>>> inversions('DCBA')
6
```