EN 001203 Computer Programming

L005: Character and String Exercise

Faculty of Engineering, Khon Kaen University

Submission: https://autolab.en.kku.ac.th

==========

* Submit an answer to a question with a file with `txt` extension. E.g., an answer for Q1 should be submitted in a text file "`Q1.txt`"

* Submit a program to a (programming) problem with a file with `cpp` extension. E.g., a program for P2 should be named "`P2.cpp`"

* All answers and programs must be packaged together in a tar file.

* Each question or problem is worth 60 points. The 240 points are counted as full score. Students are encouraged to work on as many problems as they can, but seeing 240 points could make any programming instructor super content already.

==========

"The best way to cheer yourself up is to try to cheer somebody else up."

Mark Twain

P1. Switching case. Write a program to take a letter from a user and make it a lower case if it is an upper case and vice versa. The program keeps asking a user for a letter until a user enters a non-alphabet letter (e.g., a number).

Have the program interact exactly like what shown in the examples. Letters and a number are inputted by a user.

Output example: when inputting a, B, e, G, and 1.

```
letter: a
--> A
letter: B
--> b
letter: e
--> E
letter: G
--> g
letter: 1
```

P2. Capitalize first letter. Write a program to take a word from a user and make its first letter upper case. The program keeps asking a user for a letter until a user enters a word "quit".

Have the program interact exactly like what shown in the examples. Words are inputted by a user.

Output example: when inputting n as peace, Harmony, content, spiritual, and quit.

```
word: peace
--> Peace
word: Harmony
--> Harmony
word: content
--> Content
word: spiritual
--> Spiritual
word: quit
--> Quit
```

P3. Capitalize an entire string. Write a program to take a word from a user and make every letter upper case. The program keeps asking a user for a word until a user enters a word "quit".

Have the program interact exactly like what shown in the examples. Words are inputted by a user.

Output example: when inputting n as grit, perseverance, ExPeRtIsE, Autonomy, PurPose, and quit.

```
word: grit
--> GRIT
word: perseverance
--> PERSEVERANCE
word: ExPeRtIsE
--> EXPERTISE
word: Autonomy
--> AUTONOMY
word: PurPose
--> PURPOSE
word: quit
--> QUIT
```

P4. Simple GPA. Given that grade is A, B, C, D, or F, write a program to calculate GPA from numbers of credits and grades entered. The program keeps asking for a number of credits and grade until a user enters 0 credit.

[Hint: GPA = GP/NC where NC is a total number of credits and GP = G * C when C is a number of credits and G is a grade value, i.e., A = 4, B = 3, C = 2, D = 1 and F = 0.]

Have the program interact exactly like what shown in the examples. Credit and grade are inputted by a user.

Output example: when inputting credits and grades as 3, A, 1, D, 0, and q.

```
Enter credit: 3
Enter grade: A
Enter credit: 1
Enter grade: D
Enter credit: 0
Enter grade: q
Total credits: 4
GPA: 3.25
```

Output example: when inputting credits and grades as 1, D, 1, F, 2, B, 2, C, 3, A, 3, A, 0, and q.

```
Enter credit: 1
Enter grade: D
Enter credit: 1
Enter grade: F
Enter credit: 2
Enter grade: B
Enter credit: 2
Enter grade: C
Enter credit: 3
Enter grade: A
Enter credit: 3
Enter grade: A
Enter credit: 0
Enter grade: q
Total credits: 12
GPA: 2.91667
```

P5. Simplified Triads. Write a program to calculate a simplified triad chord. (See https://en.wikipedia.org/wiki/Triad_(music) for more details.) The program takes a letter representing the first note of the chord. The letter is C, D, E, F, G, A, or B. Then, the program computes two other notes of the triad and reports all 3 notes. The program keeps asking for the first note until a user enters 'H'.

The simplified triads are (C, E, G), (D, F, A), (E, G, B), (F, A, C), (G, B, D), (A, C, E), and (B, D, F).

[Hint: seven if-statements may work, but it may be more fun treating character as a number and work the math: modulo % can be handy.]

Have the program interact exactly like what shown in the examples. Note 1 is inputted by a user.

Output example: when inputting C, G, A, B, and H.

```
Note 1: C
--> CEG
Note 1: G
--> GBD
Note 1: A
--> ACE
Note 1: B
--> BDF
Note 1: H
```

P6. Count k-mer. DNA is composed 4 basic coding elements: A, G, C, and T. In a DNA strand, there usually are short sequences with repeating DNA codes. The repeating code sequence is called k-mer. A frequently-appear k-mer is likely to have important hidden meanings. Therefore, identification of frequently-appear k-mers is the first step of studying DNA hidden meaning.

Write a program to as a user for a DNA sequence and a k-mer (a short sequence of interest), then a program counts a number of k-mers appear in the DNA sequence.

Have the program interact exactly like what shown in the examples. A DNA sequence and a k-mer are inputted by a user.

Output example: when inputting a DNA sequence and k-mer as 'ACAACTATGCATACTATCGGGAACTATC' and 'ACTAT'.

```
DNA: ACAACTATGCATACTATCGGGAACTATC
k-mer: ACTAT
--> 3
```

Output example: when inputting a DNA sequence and k-mer as 'ACAACTATGCATACTATCGGGAACTATC' and 'AC'.

```
DNA: ACAACTATGCATACTATCGGGAACTATC
k-mer: AC
--> 4
```

Output example: when inputting a DNA sequence and k-mer as 'CGATATATCCATAG' and 'ATA'.

```
DNA: CGATATATCCATAG
k-mer: ATA
--> 3
```

Notice that

1. ACAACTATGCATACTATCGGGAACTATC has k-mer "ACTAT" appear 3 times.
2. ACAACTATGCATACTATCGGGAACTATC has k-mer "AC" appear 4 times.
3. CGATATATCCATAG has k-mer "ATA" appear 3 times including the overlapping.

"I think a hero is any person really intent on making this a better place for all people."

Maya Angelou
