CSE101 – Spring 2021
Programming Assignment #3

Due April 1, 2021 by 11:59pm, KST. The assignment is worth 13 points.

Instructions

For each of the following problems, create an error-free Python program.

- Each program should be submitted in a separate Python file that follows a particular naming convention: Submit the answer for problem 1 as “Assign3Answer1.py” and for problem 2 as “Assign3Answer2.py” and so on.
- These programs should execute properly in VS Code using the setup we created in lab.
- At the top of every file add your name and Stony Brook email address in a comment.
- Include all the provided test cases in your solutions – for test cases that just return a value, make sure to add a print() statement so you can see the result.

Regarding working in pairs:

- You are welcome to work with a partner on the homework assignment, but you **MUST write both your names and email addresses in each file** in a comment. Only one person needs to submit the homework on Blackboard.
- You are only allowed to work together with one other person – larger group submissions or collaborations (beyond high-level discussions of problems, as stated on the syllabus) are not allowed.

Problem 1: (3 points)

Complete the function `translateWord`, which returns the Pig Latin "translation" of the provided word, given as the parameter `word`. The rules of Pig Latin are:

- If a word starts with a consonant and a vowel, put the first letter of the word at the end of the word and add "ay."
    - Example: happy = appyh + ay = appyhay
- If a word starts with two consonants, move the two consonants to the end of the word and add "ay."
    - Example: child = ildch + ay = ildchay
- If a word starts with a vowel, add the word "way" at the end of the word.
    - Example: awesome = awesome + way = awesomeway

Before performing any translations, change the input word into all lowercase. You may assume that the input word contains only letters.

Hint #1: use slicing and concatenation to carve up the word and rearrange the letters as needed.

Hint #2: use nested if-statements to check if a word begins with a consonant or vowel and then, in the case where the word starts with a consonant, have an inner if-statement check if the next letter is a consonant or vowel. To get you started, two variables have been created for you:

```
vowels = 'aeiou'
consonants = 'bcdfghjklmnpqrstvwxyz'
```

The idea here is that you can use the `in` operator to check whether a particular letter in the input `word` argument is a consonant or vowel. As an example:

```
if word[0] in consonants
```

Now, to test your work, a completed function named `translateSentence` has been provided to see if your own `translateWord` function works properly.

Example:

- `print(translateSentence('To be or not to be that is the question.'))`
- Output: `otay ebay orway otnay otay ebay atthay isway ethay uestionqay`

Problem 2 (2 points)

Complete the function `calculate(x, n)`, which computes and returns the value of the following sum for the given arguments, `x` and `n`.

$$\sum_{i=1}^{n}\left(2x^{i}+\frac{n^{2}}{i}\right)=\left(2x^{1}+\frac{n^{2}}{1}\right)+\left(2x^{2}+\frac{n^{2}}{2}\right)+\cdots+\left(2x^{n}+\frac{n^{2}}{n}\right)$$

Examples:

Function Call	Return Value
`calculate(3, 6)`	`2272.2`
`calculate(-2, 4)`	`53.3333`
`calculate(7, 2)`	`118.0`

Problem 3 (3 points)

Have you ever had the experience while creating an account on a website, and as you're typing in your password, a little meter pops up to tell you how "strong" your password is? Let's create our own version of that meter. Complete the function `passwordStrength`, that takes one argument, `password`, which is a string of characters consisting of lowercase letters, uppercase letters, digits and non-alphanumerical symbols. The function allocates points to `password` based on the location of certain characters in the string. The total at the end is a "score" of sorts that measure how strong the password is.

Rule	Points to Add
the **first** character is a digit	+40
a digit that is anywhere else	+25 for each such digit
the **first** character is an uppercase letter	+25
the **last** character is a lowercase letter	+15
a lowercase letter that is anywhere else	+5 for each such lowercase letter

Examples:

Function Call	Return Value
`passwordStrength('f^BAcG')`	`10`
`passwordStrength('W63UtHTuN')`	`85`
`passwordStrength('Msq08#2wGpMm')`	`135`
`passwordStrength('qHjTt9YQ')`	`40`
`passwordStrength('gw74X5I2')`	`110`
`passwordStrength('9@%B9T(jZJ')`	`70`
`passwordStrength('y*n(q%XxVp2')`	`50`
`passwordStrength('!fxus')`	`30`

Problem 4 (3 points)

You are trying to update your resume so that you can apply to summer internships. It's been a while, and you've been taking a lot of classes, so your GPA must have changed quite a bit. You log into DegreeWorks and you discover that the IT department implemented a sentient robot on the website that automatically calculates your GPA in three ways:

1. by looking at all classes (Cumulative GPA)
2. by looking at only major/SBC classes (Major GPA)
3. by looking at only major classes (Department GPA)

The robot gives you a string with the format: `'Cumulative: #.##,Department: #.##,Major: #.##'`. You want to pick the highest number out of these, and display it proudly on your resume. Since you see yourself doing this every semester, you decide to use the skills you learned in CSE 101 to automate this process with Python. Complete the function `chooseHighestGPA`, which extracts the three portions of the strings that contain the GPAs, converts them into numbers, and returns the largest of these three values.

However, you run into a problem. English is not the robot's first language, so it will misspell words. For example, it might give you the string `'Coomoolative: #.##, Depertmant: #.##,Mayejohr: #.##'`. Because of this, you decide to write your code by only looking at the numbers and commas.

Hint: the functions `split()` and `float()` could be really useful.

Examples:

Function Call	Return Value
`chooseHighestGPA('Cumulative: 3.84,Department: 3.91,Major: 3.87')`	`3.91`
`chooseHighestGPA('Coomoolative: 3.90,Depertmant: 3.75,Mayejohr: 3.85')`	`3.9` or `3.90`
`chooseHighestGPA('Cumulateeve: 4.00,Deperatmaerafsndfat: 4.00,MAYERasdf: 4.00')`	`4.0` or `4.00`

Problem 5 (2 points)

You have been given a ladder and a small robot that can climb up and down that ladder, one rung at a time. The robot can be programmed using a language that contains exactly three commands, which are always given using uppercase letters:

- `C` (climb) causes the robot to move one rung up the ladder. If the robot attempts to climb up from the top rung of the ladder, it falls off the ladder and has to start from the bottom.
- `D` (descend) causes the robot to move one rung down the ladder. If the robot is already on the first rung of the ladder, nothing happens (it stays in place).
- `J` (jump) causes the robot to immediately jump off the ladder return to the first rung.

Given a size (total number of rungs) for the ladder and a sequence of commands for the robot, your task is to determine which rung of the ladder the robot is standing on at the end of the simulation. Note that the robot always begins on the first rung (rung #1) of the ladder.

Complete the function `ladder`, which takes two arguments: the number of rungs in the ladder and a string of commands, all of which will be provided in uppercase. The function returns the number of the rung the robot ends on.

For example, suppose that you have a 4-rung ladder and the 10-instruction sequence `'CDDCCCCCJC'`. In this case:

1. `C`: The robot first moves up one rung to rung 2.
2. `D`: The robot moves down one rung to rung 1.
3. `D`: The robot attempts to move down another rung. It is already on the first rung, so nothing happens.
4. `C`: The robot first moves up one rung to rung 2.
5. `C`: The robot first moves up one rung to rung 3.
6. `C`: The robot first moves up one rung to rung 4.
7. `C`: The robot attempts to move up one rung and falls off the ladder. It starts over at rung 1.
8. `C`: The robot first moves up one rung to rung 2.
9. `J`: The robot jumps off the ladder. It starts over at rung 1.
10. `C`: The robot first moves up one rung to rung 2.
11. The function returns `2`.

Examples:

Function Call	Return Value
`ladder(4, 'CDDCCCCCJC')`	`2`
`ladder(5, 'CDDCCCCCJC')`	`2`
`ladder(5, 'CDDCCDCCDCCC')`	`1`
`ladder(10, 'CDDCCDCCDCCC')`	`6`
`ladder(10, 'CDDJCCDCCDCCJCCCDCCCCC')`	`8`
`ladder(7, 'CDDJCCDCCDCCJCCCDCC')`	`5`
`ladder(7, 'DDCDCCCDCJCCDCDCC')`	`4`