Lab 4: Strings/Loops

(55 pts, Due Apr 8 at midnight)

For this lab, you may work with a partner or you may work alone. All functions should be commented appropriately. If there are random numbers, the function must still be commented clearly, but you can use a possible results as an example of a test case.

Create a file called lab4.py. Include as comments your name, your partner's name, class time, TA's name, and lab due date. Then include and comment the following functions:

Note: Unless otherwise noted, always include comments for each function that include the input type(s), the output type, how your function calculates, and then at least two test cases. If test cases aren't possible because of random numbers or user input, then give examples of what should happen.

Part 1:

#Problem 1: (2 pts)Write a function that prints out every letter in a string, with a space between each letter.

#Problem 2: (3 pts) Given the following string: "hippopotomonstrosesquippedaliophobia" (one of the best words in the world! – it means "fear of big words!!!") write a function that takes as input the above string, and generates two random numbers between 0 and the length of the string, then returns the slice of the above word between the two random numbers.

#Problem 3: (5 pts) Write a function that takes as input 2 strings (to follow), a blank string, and an integer x. The integer, when you pass it into the function, should be an odd number greater than 1. You're going to make a random word by selecting a consonant randomly from the 2nd string, then a vowel from the first string, and continue until you have x letters. The random word you create should be returned from the function.

The strings are:

"aeiou","bcdfghjklmnpqrstvwxyz"

For example, if the integer passed in was 5, you could end up with "kidox", whereas if the integer was 7 you could end up with "lucavah"

(Side note: Surprisingly, early on we used a technique similar to this to generate "neutral words" when creating a synthetic speech database – Siri and Alexa were made by concatenating speech segments together. Because emotion and intonation in speech means the segments may not join together smoothly, one option is to create nonsense words that have no preconceived emotions or excessive intonation associated with them. But they still have to be words that a speaker can read easily.)

#Problem 4. (5 pts) When I was a kid, I used to speak a language we called "gibberish" to my friend when we didn't want adults to understand us. For this language, every vowel would be replaced with ithag and then the vowel. So, for instance,

- For instance, the word "cat" would become "cithagat"
- Monkey would become "mithagonkithagey"

Write the comments with input, output, and test cases for a function that uses a while loop to create a new word in the "gibberish" language. So every time a vowel is encountered in a word, it is replaced with "ithag",then the vowel, then the rest of the word.

#Problem 5 (23 pts total): Simple encryption. Note: read all 4 parts first in order to be able to understand each of the functions

Part a: (6 pts) You're going to write a function that takes the alphabet (as a string) and create a new, randomly scrambled version of the alphabet and returns this new random alphabet string. Each letter must only occur once in the new randomly scrambled version of the alphabet. So, for instance, my input was:

z = "abcdefghijklmnopqrstuvwxyz"

scramblefunc(z,"")

and my returned string was:

"hsyevxgrjlpntfqbzwmkadocui"

Note: just use all small letters. Let's not deal with capital letters.

*Side Note: The challenge here is to make sure that you don't put a letter into the new randomly sorted string twice. Your best bet is to check to see if z[random number you generated] is **in** the scrambled string that you're creating. If so, generate another random number. I'd suggest using a separate function for this, although it isn't necessary.*

Part b: (7 pts) Write a second function that takes as input parameters a string that is a word (e.g., "puppy", "cat"), a string that is the alphabet (the input string in part a) and a string that is the random alphabet (the string that was returned from a). Now, using the encrypted string, encrypt a word and return that encrypted word.

For this, the easiest way to do it is to use the **index** method, which is built into python.

Index returns the index of the first occurrence of a character in a string. So, for instance, using the above string z with the alphabet in it, to find the location of the character f in z, you'd do the following:

ind = z.index('f')

And ind would now hold 5

Now use index 5 with the random alphabet string to find f's translation (which, in this case, would be 'x')

So, using the encrypted string above, if the input was 'cat', the returned value would be: 'yhk'.

Part c: (5 pts)Now write a function that takes as input parameters the alphabet string and the scrambled alphabet string and the encrypted word. This function should translate in reverse. It should be pretty similar to the function in Part b. I'm going to let you figure out how to do it. This function should return the unencrypted word. If it is working correctly, the string returned from this function will be the input word string used in Part b.

Part d: (5 pts)Write a function that takes as an input parameter a word string. It:

1. prints out the string. (You can call the function you wrote for problem 1, or you can write a loop that prints out the string).
2. It then calls the function you wrote for part a, setting a variable to hold the value returned from that function.
3. It prints out the scrambled string that was returned (again, use problem 1's function or write a loop).
4. It then calls the function written for part b with the alphabet string, the scrambled string returned from part a, and the word that was the input parameter to this function. A variable should be set to hold the returned encrypted word
5. It prints out the encrypted word (held in the variable, above)

6. It then calls the function written for Part c with the alphabet string, the scrambled alphabet string, and the encrypted word. A variable should be set to hold the unencrypted word returned from this function.
7. It should print out the unencrypted word. This should be the same as the word that was input into the function and printed in 1.

Call this function at least 3 times with 3 different words of different lengths. Note that each time the encryption will be different.

Problem 6: (5 pts)Counting words:

Write a function that takes as an input parameter a string that is a sentence. It should return the count of the number of words in the string. So, for instance, if the input string was, "Houston, we have a problem!" the return value should be 5.

Problem 7: (12 pts) Write a function that takes as input parameters a string that is a sentence or paragraph, a string that is a word, and a second string that is a word. This function returns a string in which all occurrences of the first word are replaced with the second word. So, for instance, if the input string is:

"Of all my children, Sam is my favorite. Sam has always been my favorite, and Sam will always be my favorite. I know mothers aren't supposed to have favorites, and I do love all my children equally, but if I did have a favorite it would definitely be Sam."

And the first word was Sam and the second was Charlie, the returned string would be:

"Of all my children, Charlie is my favorite. Charlie has always been my favorite, and Charlie will always be my favorite. I know mothers aren't supposed to have favorites, and I do love all my children equally, but if I did have a favorite it would definitely be Charlie."

Note that there is more than one way to tackle this problem. Try to break it down into smaller problems and write small functions you could call to help. For instance, you might (but are not required) to write a function that takes the original array, an index, and the first word, and returns a Boolean value indicating if the sentence starting at that index matches the word or not. You could write another for removing a slice or replacing a slice of an array.

Extra Credit (10 pts):

Your job: get the string "123456789" to sum to 100, by inserting in the string two minus symbols and one plus symbol (in any order) to make the formula correct. You aren't allowed to re-arrange digits, or do some toothpick math like combine two minus signs to make a plus. You must use every digit, and all three operators. For example, here's one incorrect solution:

123 + 45 - 67 - 89 = 100 (This is an incorrect formula; it totals 12)

You want to write a function that does the thinking for you. It should output every possible equation that can be formed, and the actual result of that equation. The equation(s) that result in 100 should have stars around them. At the end, you should print out the number(s) of formulae that were possible. Here's an excerpt of some example output:

- ...
  1 -2- 3+456789 = 456785
  1 -2+ 3-456789 = -456787
  1 +2- 3-456789 = --456789
  …

- 123456 - 7 - 8 + 9 = 123450
  123456 - 7 + 8 - 9 = 123448
  123456 + 7 - 8 - 9 = 123446
  ...
  168 possible equations tested

Hint: This problem heavily relies on loops and slicing.

#Problem 5: (5 pts) Many digital identification numbers such as ISBNs, credit card numbers, bank routing numbers, and UPCs, and even routing packets, are designed with an extra bit, known as a parity bit, that helps detect (and sometimes correct) human entry errors. In this problem we will modify this a bit to assume a simple check-digit algorithm:

Parity will be determined by adding all the digits in the number up, minus the last digit (the right-most digit). If the digits add up to an even number and the right-most digit is a 0, then the number is most likely okay. If all the digits add up to an odd number and the right-most digit is a 1, then most likely the number is okay. If the digits don't match the parity number, then the number has probably been corrupted. If the parity number is anything other than a 1 or a 0, the parity number has been corrupted, Otherwise, as far as we know, the number is okay.

Write a function (including comments) that uses a while loop to check the number's validity using the above checking system. This function will return a Boolean value.

#Problem 7: (2 pts) A teacher may want students to practice their multiplication tables. Each week it's a different number. So for instance, one week the students should practice their 4s multiplication tables, and the next week their 6s multiplication tables.

Write a function that takes as an input parameter an integer. The function then generates a random number between 0 and 12 including 12. The function then uses the **input command** to ask the user, "what is randnum times x?" where randnum is the random number generated, and x is the input parameter. So, for instance, if the input parameter was 5, and the random number generated was 7, the user would be asked, "what is 7 times 5?"

The function should then take the answer the user entered and check whether the user was correct or not, returning a Boolean value with True if the user was correct, and False if the user was wrong.

7b. (5 pts) Now create an online test. This test function will take as an input parameter an integer the test_num) representing number you are testing and an integer representing the number right (num_right) the user must get before they can quit the test. In the function, you will call the function you created above inside a loop. The loop will keep track of the number of times the student answered correctly (e.g., True is returned from the above function). If the student answers num_right correctly in a row, the function returns a string saying "Congratulations! You finished!" If, however, the student gets one wrong before getting num_right correct in a row, the count starts over at 0. Thus, there is only one way to get out of the loop, and that is for the user to get the answer correct num_right times in a row.