CSE101 – Spring 2021
Programming Assignment #5

Due April 29, 2021 by 11:59pm, KST. The assignment is worth 12 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 "Assign5Answer1.txt" and for problem 2 as "Assign5Answer2.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 **and write 1 additional test case** of your own for all problems that do not take user input.

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: (4 points)

For the following, copy the questions into a text file and write your answers for each question. Name the text file Assign5Answer1.txt

I. Place the following algorithm time complexities in order from fastest (least number of comparisons) to the slowest: $nlogn$, n, 2^n, n^2, $logn$, $2n$

II. Why are divide-and-conquer algorithms often very efficient in terms of time complexity?

III. Write in your own words 3 different examples of cases where people around you are giving up their privacy to use some service in exchange for a benefit. State for each example what data is being collected and what benefits the user receives in exchange.

IV. In your own words, explain the two characteristics that a recursive solution must have.

Problem 2: Scrabble Sort (3 points)

Write a Python program containing a function named `scrabble_sort` that will sort a list of strings according to the length of the string, so that shortest strings appear at the front of the list. Words that have the same number of letters should be arranged in alphabetical order. Write your own logic for the sort function (you may want to start from some of the existing sorting code we studied). Do NOT use the built-in sort function provided by Python.

One optional way to test your function is to create a list of random words by using the `RandomList` function in PythonLabs and passing `'words'` as an argument. For example:

```
>>> from PythonLabs.RecursionLab import *
>>> a = RandomList(20, 'words')
>>> scrabble_sort(a)
>>> print(a)

['mom', 'gawk', 'tree', 'forgo', 'caring', ... 'unquestioned']
```

Note: you do not need to include the above test case in your solution, but be sure to include at least 2 test cases to verify your `scrabble_sort` function works.

Problem 3: Recursive functions (5 points)

For this problem you must write the functions in a recursive manner (i.e. the function must call itself) – it is not acceptable to submit an iterative solution to these problems (i.e. there should be no `for` or `while` loops).

A. Complete the recursive function `gcd(m, n)` that calculate the greatest common denominator of two numbers with the following rules:

```
# If m = n, it returns n
# If m < n, it returns gcd(m, n-m)
# If m > n, it returns gcd(m-n, n)
#
def gcd(m,n):
    return None  # Replace this with your implementation
```

B. Complete the following function that uses recursion to find and return the max (largest) value in the list u.

```
# find_max([1, 7, 4, 5] returns 7
# find_ max ([1, 7, 4, 5, 9, 2] returns 9
#
def find_max(u):
    return None  # Replace this with your implementation
```

C. Complete the following recursive function that returns the zip of two lists u and v of the same length. Zipping the lists should place the first element from each into a new array, followed by the second elements, and so on (see example output).

```
# zip([1, 2, 3], [4, 5, 6]) returns [1, 4, 2, 5, 3, 6]
#
def zip(u, v):
    return None  # Replace this with your implementation
```

D. Complete the following recursive function that removes all occurrences of the number `x` from the list `nums`.

```
# remove_number(5, [1, 2, 3, 4, 5, 6, 5, 2, 1]) returns [1, 2, 3, 4, 6, 2, 1]
#
def remove_number(x, nums):
    return None  # Replace this with your implementation
```

E. Write a recursive function `removeLetter(string, letter)` that takes a string and a letter as input, and recursively removes all occurrences of that letter from the string. The function is case sensitive.

Some example test cases are below:

```
>>> removeLetter("test string", "t")
es sring

>>> removeLetter("mississipi", "i")
mssssp

>>> removeLetter("To be or not to be is a question.", "t")
To be or no o be is a quesion.
```