CISC121 Winter 2023 - Assignment 4

Due: Wednesday March 15th, 2023 (11:59pm)

Notes:

- Your assignment should follow the python style guide, and any other style/commenting expectations as discussed in lecture
- By submitting the assignment, you agree that you have followed the Queen's Academic Integrity policy, and that your assignment was completed independently.

Scenario

The following algorithm describes a method of checking if two strings are anagrams of each other. We first describe the algorithm.

> First, assign each uppercase character a unique prime number[1]. Let *word* be some string of characters of only uppercase letters. We can compute the *word_value* by taking the product of each prime corresponding to each letter.
>
> Second, suppose we have two different strings *word1* and *word2*. If the product of primes for word 1 and word 2 are equivalent, then these strings must be anagrams of each other.

In this assignment you will write a recursive function that computes the prime product of a string using the unique mapping function as described above. In order to do this, you will first need to map each letter to a unique prime number. Then, you will write a recursive function that computes the product of primes for some given string. Finally, you will put this altogether in your main program to take two strings from the user, and determine if they are anagrams of each other by calculating their prime products.

[1] 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101

Question 1 (15 marks)

In your *functions.py* file, write the following functions:

- *char_prime* will map each letter to a unique prime number by writing a function that takes a single character and outputs the unique prime that you assign to that character. [HINT: you should hard code the mapping between letters and primes in this function so that your map is consistent].
- *Primeify.* A **recursive** function that computes the product of the character primes for a given string. This *must* be a recursive function! You can use the previous function (char_prime). Consider how to build the prime value for BANANA if you already know the prime function for NANA
- *is _anagram*. This function takes in two strings, and computes their product of primes by 'primeify'ing each string. If the values are equal, then they must be anagrams.

In your main function, called *a4_q1.py* Ask the user for two strings that contain only upper-case letters. You should confirm that these are in fact strings, and should check this input. However, you may choose to give an error, or clean the code however you wish. There are no requirements here other than that you make certain the user input is a valid string of only upper case letters with no spaces. Then, call the function *is_anagram*, and give the user a message based on this output.

Note – the autograder will *only* be calling from your functions file, it will *not* be calling your main function. Be sure your functions meet the required specifications.

The docstrings for all functions are at the end of this document.

Question 2 (10 marks)

Using the strings we entered from Question 1 (ie. string1 and string2 that may or may not be anagrams), we notice that we have both a string or a list of primes. We will now sort this string, by applying a new sort to the list of primes. (Be sure you create this list of primes if it does not already exist).

Radix sort is a recursive sorting algorithm that segments the lists into "buckets." This can be done in a variety of capacities, but the most common is through position within an integer.

Read the guide listed here on Radix Sort**.**

You should have the following functions:

- *radix_sort* the radix sort

- *A4_q2.py* This should determine the string required, and formatting the list of primes. Then it will call the radix_srot on this list. A good user message should be presented once the list is sorted.

To Submit:

Submit the following files as a **SINGLE zip file** named **a4_999999999.zip (your student number)**:

1. **functions.py**: A file containing all of your functions for both questions.
2. **a4_q1.py:** the main program for the first question
3. **a4_q2.py:** the main program for the second question
4. **testing.txt:** Include all tests you run, and the output. Simply state what the resulting set of input was, and then the corresponding output. Whenever your tests fail, you may include notes to the TAs. It is encouraged to discuss *why* you've chosen each test case, but not required. It is good to have 3-5 tests for each function. You should be testing your code much more often than this.

Appendix – Docstrings

```
def char_prime(my_char):                                                    1
  """                                                                       2
  -------------------------------------------------------                   3
  Converts an uppercase letter to a unique prime number                     4
      Based on the conversion given in the footnote                         5
  -------------------------------------------------------                   6
  Parameters:                                                               7
      my_char - a char in ABCDEFGHIJKLMNOPQRSTUVWXYZ (char)                 9
  Returns:                                                                  10
      prime_int = a prime number unique to the letter                       11
  -------------------------------------------------------                   12
```

```
def primeify(my_string):                                                    1
  """                                                                       2
  -------------------------------------------------------                   3
  RECURSIVELY gives the product of primes corresponding to the letters      4
            in the string                                                   5
```

```
  ------------------------------------------------------
  Parameters:
      my_string - any string (str)
  Returns:
      prime_product = the product of all primes for each letter
  ------------------------------------------------------
```

```
def is_anagram(string1,string2):
  """
  ------------------------------------------------------
  Determines if two strings are anagrams of each other

  ------------------------------------------------------
  Parameters:
      string1, string2 - any two strings (str)
  Returns:
      is_anagram = whether or not they are anagrams (Boolean)
  ------------------------------------------------------
```