Programming Assignment 1

Random Number Chaining

CECS 328 – Fall 2023

Due: Friday, September 15 at 11:59 pm

Points Possible: 10

Problem Statement:

In this assignment, a comma separated values (CSV) file contains two columns. Each column contains randomly generated positive integers. The value in the second column of a row links to the value of the first column in a different row if, and only if, the values are the same. The statement is true the other way around too: the value in the first column of a row links to the value of the second column in a different row if, and only if, the values are the same.

<u>For example:</u>

```
1551,283
283,459
```

The “283” in the first row links with the “283” in the second row because they’re the same number. It will produce a link [1551, 283, 459]. In this case, the link is of length 3.

<u>Another example:</u>

```
283,459
1551,283
```

Will produce the same resulting output as the example above.

Note that the numbers are completely random and hold no importance other than determining whether the rows can be linked.

The CSV file may contain a link that’s a cycle. However, the CSV file will not contain a scenario where a number is listed more than once in a column. In other words, a number can be listed at most once in column 1, and it can be listed at most once in column 2.

Given the above mentioned CSV file, output the longest link and the link length. The format of the output should match exactly with what’s shown in the “Examples” section below. In other words:

<chain array> + " with length " + <chain array length>

The link should be an array, with each number encased in single quotes (see “Examples” section below). The link length should be an integer (however, to output it in Python, you may need to convert it to a string).

Regarding libraries, you may only have the following import statement in your code:

```
import csv
import sys
```

The input CSV file should be located in the same directory (aka folder) as your code (pa1.py). The input CSV filename should be able to be passed in via the command line. See the "Examples" section below.

Submission Requirements:

Failure to follow any of the requirements in this document (items listed below or above) will result in point deductions, up to and including receiving zero credit.

1. Write your name as a comment at the top of your code.
2. The filename of the code that you submit should be: **pa1.py**
3. Your program must be able to be executed via the Python command line (see the "Examples" section below).
4. You can only import the following Python libraries:

   csv
   sys
5. You can work with other students or individually, up to you. However, you must submit your assignment individually on Canvas.
6. You must submit your file on Canvas. Any other submission method (such as email) will be rejected and you will receive zero credit. As mentioned in the syllabus, only submissions on Canvas will be accepted. If you forget to submit and your file is in some other location (e.g. GitHub) instead, I still will not accept it (no exceptions).
7. The programming assignment must be in the Python version 3.11 programming language or later.
8. Do not put your Python functions in a class (e.g. "`class Solution:`"). This will prevent our testing script from executing on your program and you will end up getting a zero. You will get zero credit if you do this.
9. Do not have any print statements in your code other than what is indicated in the "Problem Statement" above and "Examples" section below. If you do, your program will fail the test cases and you will get points deducted, up to and including receiving zero credit on your assignment.

Examples:

The below are run from the Python command line. This is how your code will be graded. Your program absolutely needs to be able to be run from the command line, otherwise you will get zero credit.

Two sample input files are provided to you on Canvas, so that you can test your code against. Note that getting the same output as the below will not guarantee full credit on this assignment, as these are just test files. You can create your own test file(s) to test out various scenarios. It is up to you to make your code as robust as possible.

```
>python pa1.py input-pa1-small.csv
['1', '2', '3', '4', '5'] with length 5

>python pa1.py input-pa1-large.csv
['6346', '4170', '711', '296', '5927', '2700', '5871', '2891', '4379', '2972', '708', '6529', '8971', '2837',
'1790', '253', '6831', '5972', '9930', '5227', '1041'] with length 21
```

Constraints:

- Values in the CSV file are positive integers, between 1 and 11,000.
- Input CSV file may be up to 5,000 rows.
- Your code must complete within 5 seconds.

Grading Guidelines:

- Does the program meet the requested requirements/criteria?
- Are the submission instructions followed?
- Does your code compile and execute?
- Does your code pass my test cases? I will not share my test cases with you. However, I have provided examples, as mentioned above, so that you can test your code against.