ANKARA UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

Computer Programming 1

Fall 2020-21

LAB3 Quiz
Assoc. Prof. Dr. Hacer YALIM KELEŞ

Date: 13/11/2020

Write a Python program which takes a sequence of inputs that contain multiple *source strings* and multiple *query strings*; prints the occurrences of query strings within the source strings. The output format is specified below. Please also look at the sample I/O files.

Specification details:

- Source and query strings will be provided in two different lines. The number of strings are not fixed, i.e. can change in different runs.
- You can assume that there will be at least one source string and one query string.
- If there are *no occurrences* of the query strings within any source string, you will print **None**.
- You will be printing the occurrences in the order that checks for each source sting, all the query strings. (Check your results with the sample outputs for compatibility of the order)

Constraints:

- You are *not allowed to use* ***any*** string class functions, such as **str.find()** or similar**.** You are expected to implement your own algorithm by iterating over the string content as in the previous lab assignment.
- You are *not allowed to import* any modules that helps identification of substrings as well. We will go over the source files one by one and if we identify any external library functions in your source codes, we will treat your submission as cheating and you will get 0.

Hints:

This time, since we have recently learned lists, you will read the strings given in a line using:

```
inputs = input().split()
```

After reading, ***inputs*** variable is bound to a list of strings. You can iterate over the individual items using **in** operator in a for loop, i.e. **`for item in inputs:`**.

You are expected to implement your own solutions to determine every occurrences of each query strings within the source strings. As we have seen, you can iterate over individual characters of a

string, ***str***, using **`in`** operator; e.g. **`for chr in str:.`** Alternatively, you can use len() function and range() function in iterations within strings.

Since there will be multiple nested loops, it would be a good practice to define a function that identifies the substring relation and other necessary information, given two strings. You will be able to construct your solution easier and cleaner that way. It will also help you to identify possible sources of errors easily during development. Otherwise, you may have difficulty in time management.

I personally suggest that, first write this function and test it providing different string pairs. After making sure that your substring identification function works without problems, you can very efficiently implement the parts that integrates input reading and output formatting etc., in quite a short amount of time. In every new step, such as reading the inputs part, check if everything is fine by monitoring the content that you read (like printing temporarily or debugging etc.).

I/O Format:

Input format:

[[string][Space]]$^{+}$ [Newline] **[**[string][Space]]$^{+}$

Output format:

Repeat for each occurrence i:
<source_string_i>[space]<query_string_i>[space]<total_number_of_occurrences_in_source_i>[space]<start_indexes_list>[Newline]

if no occurrences are found: None

Note that:

1- Sample input and output files are provided.

2- [x]$^{+}$ means, one or more elements of x

Testing:

You are provided with some sample I/O files. Assume that input.txt stands for a sample input and output.txt stands for the corresponding output file and your source code is named as Lab3.py; you can test your program from the command line using the following commands. (>: stands for command prompt)

> python Lab3.py < input.txt > my_output.txt

This command redirects the inputs in the input.txt file as if they are provided from the command line to your python code. The outputs, which you generate using the print() function in your source codes, are redirected to my_output.txt file.

You can then check if my_output.txt file is exactly the same with the provided output.txt file, using diff command from the command prompt:

> diff output.txt my_output.txt

Submission:

1- Name your Python source file as <student_id>.py; replace <student_id> using your student id number.

2- Upload your python file using the interface provided in e-kampüs course page.