CSL 201 Data Structures
Lab 6 Due on October 30, 11.55pm

Instructions

- You are to use Java as the programming language. Use Eclipse as the IDE.
- You have to work individually for this lab.
- You are not allowed to share code with your classmates nor allowed to use code from the internet. You are encouraged engage in high level discussions with your classmates; however ensure to include their names in the report. If you refer to any source on the Internet, include the corresponding citation in the report. If we find that you have copied code from your classmate or from the Internet, you will get a straight F grade in the course.
- The submission must be a zip file with the following naming convention - rollnumber.zip. The Java files should be contained in a folder named after the question number.
- Include appropriate comments to document the code. Include a `read me` file containing the instructions on for executing the code. The code should run on institute linux machines.
- Upload your submission to moodle by the due date and time. Do not email the submission to the instructor or the TA.

This lab will improve your understanding of pattern matching using suffix tries.

1 Pattern Matching using Wildcard Characters on Suffix Tries (25 points)

Given a long text T containing English characters, spaces and punctuation marks, Construct a suffix tries for the text T. You can implement the $O(n^2)$ algorithm discussed in the class. Find all occurrences of a given pattern P in T. The output of the program should mention the begin and end indices of all occurrences of P in T.
We will add a small twist to the problem by adding wildcard characters to the pattern. Specifically we consider the following two cases

1. Any character wildcard - `?`. This wildcard can match with any character. For example consider the following text and pattern
   $T =$ `This is an easy programming assignment`
   $P =$ `?n?`
   The output will contain indices in the text corresponding to three character substrings with `n` in the middle. In this example, these substrings are `an` , `ing` and `inm`.

2. Any sequence wildcard - `*` This wildcard can match with any sequence of characters. For example consider the following text and pattern
   $T =$ `This is an easy programming assignment`
   $P =$ `g*m`
   The output will contain indices in the text corresponding to all substrings that begin with `g` and end with `m`. For our example, these substrings are `gram`, `gramm`, `gnm`, `g assignm`, and `gramming assignm`.

We will assume that the wildcard characters will not be part of the input text and that the pattern P will contain at most one wildcard character `*`.

Input Format

The input text and patterns should be read from a text file. The text file will first contain the input text, followed by a number indicating the number of patterns to be matched. The next n lines will define the pattern. For example

```
This is an easy programming assignment.
4
is
m*n
a?
```

Output Format

The output is the begin and end indices of all patterns matching with pattern one (one per line) followed by those of pattern two and so on. The patterns should be sorted on the begin index followed by end index. Assume that the index of the text starts from 0. For the previous input example, this will be the output.

```
2 3
5 6
22 25
22 33
22 36
23 25
23 33
23 36
34 36
8 9
12 13
21 22
28 29
```

2 Documentation of Code (5 points)

Documentation of code will fetch 5 points.