Lab Exam 3 (19:10-20:25)

String Matching (100 pts)

In this lab, you are going to implement the string matching algorithm with finite automaton. You will be given a text T

and a pattern P, your task is to find occurrences of pattern P on text T. Details are given as follows:

1. The alphabet consists of only lower case alpha characters. Namely, S={a, b, ..., z}.
2. The length of the text is always greater than or equal to the length of the pattern. Namely, |P| <= |T|.
3. P may contain a special character which is indicated by character '?'. P can contain at most one special character and it cannot be the first character of P. While searching for P, if P contains the special character, the character preceding '?' may omitted, i.e 'ba?' matches following strings ,'ba' and 'b'.
4. You will be given two characters, which will be treated as same during matching. For example, if characters 'a' and 'b' given, then the pattern 'cab' matches following strings, 'caa', 'cab', 'cba', and 'cbb'.
5. P may occur more than one time, even worse, they may overlap. You have to find all occurrences of P's in T (Please take a look at the sample inputs & outputs section to better understand this case.).
6. |T| <= 1.000.000 and |P| <= 20.
7. You will return vector of unsigned integers representing starting point of each occurrences of P. If P does not occur in T, you should return empty vector. **Please note that first character of the text is position 0, not position 1.**
8. You have to implement string matching algorithm with finite automaton. Other string matching algorithms will be graded as 0 even they work correctly. However, algorithm to construct the finite automaton is up to you (We do not expect the most efficient finite automaton construction algorithm, complexity will be fine.).

Input: Text (char *), Text size (unsigned integer), Pattern (char *), Pattern size (unsigned integer), Character 1

(char), Character 2 (char)

Output: vector of unsigned integers

Signature: vector<unsigned> find_occurrences(char * T, unsigned text_size, char * P, unsigned pattern_size, char

c1, char c2);

Sample Inputs & Outputs

Sample Input 1

```
T: abcabcabc
text_size: 9
P: ab
patter_size: 2
c1: a
c2: d
```

Sample Output 1

```
0 3 6
```

Sample Input 2

```
T: abcabcabc
text_size: 9
P: cb
patter_size: 2
c1: a
c2: d
```

Sample Output 2

```
Empty
```

Sample Input 3

```
T: abcabcabc
text_size: 9
P: cb
patter_size: 2
c1: a
c2: c
```

Sample Output 3

```
0 3 6
```

Since 'a' and 'c' are given as equals, the pattern 'cb' matches both 'cb' and 'ab'.

Sample Input 4

```
T: abcabcabc
text_size: 9
P: cb
patter_size: 2
c1: c
c2: b
```

Sample Output 4

```
1 4 7
```

Since 'b' and 'c' are given as equals, the pattern 'cb' matches following strings;

- 'cb',
- 'cc',
- 'bc',
- 'bb'.

Sample Input 5

```
T: abcabcabc
text_size: 9
P: c?b
patter_size: 3
c1: a
c2: d
```

Sample Output 5

```
1 4 7
```

The pattern 'c?b' matches both 'b' and 'cb' due to special character '?'.

These sample inputs are given only for visualization purposes. You will implement a function and receive inputs as parameters and return the integer as the output.

Specifications

- **Programming Language:** You must code your program in C++. Your submission will be compiled with g++ on moodle.
- **Requested Files:** You will only be graded on "lab3.cpp" file.
- **Library:** You are allowed to use any standard C++ library in your implementation with the exception of string/cstring libraries. Cheatsheets are provided on course home page.
- **Testing:** You can test your code using run functionality on the editor. It will compile and execute "test.cpp" file.
- **Evaluation:** Black box evaluation method is going to be used. Inputs given to you during your lab exam is for testing your code and will not determine your grade.
- **Cheating:** In case of cheating, the university regulations will be applied.
- **Leaving:** In the first lab session, students are not allowed to leave the lab until the end of the exam.
- **Grading:** Each task will be evaluated based on the correctness.