CSE3080 Data Structures (Spring 2022)
Homework 1: String Pattern Matching

Handed out: Mar 24, Due: April 12, 11:59PM (KST)

1. Problem Description

In this homework, you are going to implement a program which accepts two strings (**str**, **pat**) and checks whether **pat** appears as substrings in **str**.

In the lectures, we saw program **ex016** and **ex017** that use the naive algorithm and the KMP algorithm for string pattern matching, respectively.

In this homework, we are going to change the problem slightly. If pat appears in str multiple times, the program should print all starting positions and count the number of appearances.

Let us consider the following example. Suppose the two strings **str** and **pat** are:

```
str: hello world my name is world.
pat: world
```

We can see that the pattern "world" appears twice in the string. Your program will output the following two lines.

```
2
6 23
```

The first line indicates the number of appearances of the pattern in the string. The second line indicates the starting positions in the string where the pattern appears. Note that the first character of the string is **index 0**. Multiple starting positions are **separated by a blank (' ')**.

Let us consider another example. Suppose the two strings str and pat are:

```
str: aaaabaaaa
pat: aa
```

Then, your program should output the following two lines.

```
6
0 1 2 5 6 7
```

You are going to write two versions of the pattern matching program. The first one uses the naïve algorithm, whereas the second version uses the KMP algorithm.

2. Your task and requirements (Read Carefully!)

(1) You should write two C programs, named **hw1_naive** and **hw1_kmp**. The first program uses the naïve algorithm which has the time complexity of O(mn), where m is the length of the string and n is the length of the pattern. The second program uses the KMP algorithm which has the time complexity of O(m+n).

(2) In addition to the program, you should also write a **Makefile**. The TA will build your code by running "make". It should create the binary files, **hw1_naive** and **hw1_kmp**.

(3) Your program should read the inputs from files and output the result to files. The input files are "**string.txt**" which contains the string, and "**pattern.txt**" which contains the pattern. Your output file should be named "**result_naive.txt**" and "**result_kmp.txt**".

(4) You can generate example inputs using the attached program probgen. It creates a string of 9,999,999 characters and a pattern of 2,999 characters. Your program should support this long string, so consider this and make your arrays large enough. Your program does not need to support string and pattern sizes that are larger than these.

※ If you declare a large array as a local variable inside a function, your program might generate a segmentation fault. To avoid segmentation faults, declare large arrays as global variables.

(5) The format of the input files is simple. Both string.txt and pattern.txt contain a single line of string. You can easily read them using functions like **fgets**.

(6) The format of the output files is as described earlier. The output file **result_naive.txt** and **result_kmp.txt** should contain two lines. The first line is the number of appearances of the pattern in the string. The second line outputs all starting positions in the string where the pattern is found.

(7) If the input files do not exist on the same directory as your binary file, you should output an error message and end the program. An example error message could be:

```
The string file does not exist.
```

(8) You will be given example binary files of hw1_naive and hw1_kmp. Try running them with input files created using **probgen**. See how the two programs produce the same outputs, but there running times are different. Your program should have similar running times with the given examples, and exactly produce the same outputs as the given examples.

3. Submission

You should submit your source codes and the Makefile. (You do not need to submit compiled binary files.) Combine your files into a zip file named cse3080_mp1_**20200001**.zip. The red numbers should be changed to **your student ID**. Submit your files on the cyber campus.

4. Evaluation Criteria

(1) Your two programs should compile and produce binary files.

(2) If the input files do not exist, the program should display an error message and terminate.

(3) If the input files exist, your two programs should produce correct answers, not only on the given example inputs but in all cases. You can assume that the size does not exist that of the given example.

(4) hw1_kmp should notably run faster than hw1_naive for the given example inputs. Your programs should show similar running time compared to the given example binary files.

5. Notes

- You must write your own code. You may discuss ideas with other students but must not copy their work. Similarly, you can find ideas on the Internet, but must not copy the code from the sources obtained from the Internet.

- Our department uses a software that detects duplicate codes. Since the software uses an assembly code level detection, just changing the variable names will not bypass the software. Make sure you write your own code from the scratch. Then, you will have no problem.

- Your program should be able to compile and run on the **colab**. Make sure you check before you submit your work. The TA will download your code, run “make” and execute your binary files to check if they produce correct outputs.

6. Late Policy

10% of the score is deducted for each day, up to **three days**. Submissions are accepted up to three days after the deadline.