CSC 212—Algorithms and Complexity
Implement the Knuth-Morris-Pratt (KMP) pattern-matching algorithm using DFAs.

Practical 4 Term 3 3 August 2015

Submit as soon as possible before leaving the practical.

Implement the Knuth-Morris-Pratt (KMP) pattern matching method using a DFA as described by in Sedgwick and Wayne (S&W) on pages 762–769.
Use the the same data as for last week's practical.

Copy your previous working practical in `33practical` into today's directory `43practical`. Implement the Knuth-Morris-Pratt (KMP) pattern matching method using a DFA. Learn how to build the DFA first. Check your if your code produces the same DFA for `"ABABAC"` as that given on page 767 of S&W. Next construct the DFA for the pattern `"xyxyyxyxyxx"` and compare it with its `next` table on page 298 of the class slides. Finally use the large set of words in the file

`/export/home/notes/ds/lesmiserables.text`

in the SunLab as the file in which you must look up and count appearances of these words that we call patterns. A supply of such patterns to be used is stored in the file `patterns.text` in the same directory.

First the entire text file must be read into the array `String T` and then the patterns must be read *one-by-one* into the variable `String p`. Each pattern must be looked up using the new version of KMP. The final output of your code to the screen is the number of patterns whose first occurrence it has matched within 1 second.

We advise you to test your code by checking if it works for the first 10 (ten) patterns. When this works, proceed to do the timing. If you read any ambiguity into this problem statement please resolve it yourself.

1. *Preparation of file system*: Today's practical must be done as usual inside your name file that lies in your home directory. Do today's work in a directory called `43practical`. Create the new directory by copying your `33practical` directory into the new one. Fix up the `Makefile` so that it processes today's work.

2. **Today**: Re-create a Knuth-Morris-Pratt (KMP) pattern searching method that works for a fixed piece of text called `T` and smallish but fixed patterns called `p` using DFAs for the patterns.

   Try to understand the code rather than just typing it from the textbook. Your understanding is tested during evaluation, not your typing skills. Your coding skills will improve if the code is always your own. Do your debugging by first getting each method to work with a small text `T` and pattern `p` that are assigned in the test code before trying to apply the search methods on the larger text and patterns from the files. Also do tests that (1) succeed at the start of `T`, (2) somewhere away from the end points of `T`, and (3) one that matches at the end of `T`, for example:

```
String p = "The";
String T = "The Rabin-Karp method is much faster that the brute-force method.";

String p = "brute-force";
String T = "The Rabin-Karp ethod is much faster that the brute-force method.";
```

```
String p = "method.";
String T = "The  Rabin-Karp method is much faster that the brute-force method.";

String p = "";
String T = "The  Rabin-Karp method is much faster that the brute-force method.";

String p = "method.";
String T = "";

String p = "";
String T = "";
```

Please note that these examples are textual, but the actual practical uses strings that consist entirely of strings separated only by spaces or newline characters—other characters are considered as parts of the words. Once you are convinced that your code works for these examples the program must be enhanced to read a pattern `p` from the `patterns.text` file and search for that pattern once in the text `T` that has also been read from a file. *Do not copy the data.* Simply read it and use it to generate your results. The data is stored in the files given above.

- Time the new KMP method for 1 second and report the number n of patterns that were searched.

3. Submit ALL your pracs in a file that has been created in the usual manner: i.e., `cd; make submit`.

4. Note that all the occurrences of a pattern must be looked up and counted.

5. In summary look up each "pattern" `p` throughout the entire "text" file `T`.
