CSCE A351: Automata
Spring 2020. Homework Assignment 3
Due: 04/24/2020 11:59AM AKDT
Individual assignment

Provide code in a programming language you choose so you can make your point in terms of average and worst-time complexity for the following problems. Provide a write-up analyzing the performance of your code in average and worst-case scenarios as well.

1. Implement naive multi-occurrence string search: given a string, in the form of a data-structure that allows each of the characters of the string to be accessed in $\mathcal{O}(1)$, called the haystack, and another string called the needle, the function computes all indices i such that the needle appears as a substring of the haystack, starting at the i-th character of the haystack. Return the set of such indices i in the form of an array of booleans.

   In C, your function could have the following signature:

   ```
   void matchStrings(char *res,
                     const char *haystack,
                     const char *needle);
   ```

2. Implement Rabin-Karp multi-occurrence string search. This function does the same thing as the naive multi-occurrence string search specified above but uses Rabin's and Karp's sliding-window hashing and hash-update technique, which we discussed in class. If you use C as a programming language, you may start your code from the boilerplate code provided on Blackboard.

3. Implement Knuth-Morris-Pratt multi-occurrence string search. This function does the same thing as the naive multi-occurrence string search specified above but uses the Knuth-Morris-Pratt implicit finite-automaton technique described in the textbook.

4. Implement a test driver. If you use C as a programming language, you can utilize the test driver posted on Blackboard.

5. Run your three different implementations on various examples, showcasing their average and worst-case complexity.