Objective: To gain an understanding of empirical analysis of algorithms and algorithm visualization.

Background: Read Sections 2.6 and 2.7 of the text where techniques of empirical analysis and visualization of algorithms are described. Another viewpoint can be found in "Experimental Analysis of Algorithms" by Catherine McGeoch (Notices of the AMS, Volume 48, Number 3). Refer to Section 3.2 of the text for the pattern matching algorithm.

Programs and Data: It is recommended that you completely debug each of the following before proceeding to the next:

1. Write a program to implement the brute-force pattern-matching algorithm. Your program should work with patterns of arbitrary length (up to 100 characters) and texts also of arbitrary length (up to 10000 characters).
2. Create a new version of your program that visualizes the operation of the algorithm. You can be as creative as you wish here. The idea is to show the pattern and text and how they are lined up in each step of the algorithm and to give visual clues as to the character comparisons performed. This only needs to work well for relatively small input sizes.
3. Create (pseudo)random patterns and texts of bits (0 or 1) using either the pseudorandom number generator that is available through your system or implement your own using the instructions in section 2.6. You will need patterns of lengths 10, 50, 100, and texts of lengths 1000, 2000, 4000, 6000, 8000, 10000.
4. Create the worst-case input patterns and texts of the above lengths.
5. Create a new version of your pattern-matching program that also determines the cpu-time used. You may find that the computer you are using is too fast to register anything but 0 seconds, even for large patterns and texts. In that case repeat each pattern-matching operation a large number of times so that you can get some meaningful results.
6. Create another (separate) version of your pattern-matching program that incorporates a counter that computes the number of basic operations performed by the algorithm.

Experiment with your programs and collect statistics. In particular, you should be looking for the answers to the following questions:

- What is the average-case behavior of this algorithm?
- How many bits of the pattern are compared to bits of the text, on average, at each position where the pattern is aligned?
- What is the formula for the average number of comparisons, as a function of $n$ (the length of the text)?
- What is the formula for the average number of seconds?
- Verify that the formula for the worst-case number of comparisons is correct using the worst-case input generated.

Report: This is the main part of this assignment. Here you need to present your programs, data, and experimental results in a clear, convincing, and complete manner (remember the 3 c’s). There should be an introduction explaining the purpose of the assignment (you can borrow from this handout) and how you carried it out. The presentation of the programs should leave no doubt that they work well by showing sample runs. The presentation of the experimental results should demonstrate your understanding and include some graphs plotting the number of comparisons vs. text length (keep pattern length fixed). A conclusion should summarize the results and discuss any further observations that you made and questions that you would like to see answered or further improvements that you would make if you had more time to experiment with pattern-matching programs.