Design an experiment to compare following three string matching algorithms:
1. Brute-force string matching,
2. Horspool's algorithm,
3. Boyer-Moore algorithm.

You should implement all the above algorithms in any programming language.
The input text will be a long HTML file (longer than 1MB) and a pattern. The output will be highlighted HTML file and the number of occurrences. Your algorithms will search all the occurrences of the pattern in the HTML file and highlight these occurrences by using <mark></mark> tag and save the updated file. You are expected to print the bad symbol table and the good suffix table for each pattern. You are also expected to provide the number of character comparisons and running time (in terms of milliseconds) for all the three algorithms. You should provide extensive comments on your results.

You should work on two types of HTML files.
Type 1: HTML files with English text. You may find several long HTML files on the Internet. Patterns would be some meaningful English words.
Type 2: HTML files that include random bit-strings. You may generate these files. Patterns would be some random bitstrings with various lengths as well.
From each type, please provide at least 3 samples.

The homework has three main steps:
Step 1: Designing the Experiment: This step includes:
(a) Deciding on / generating long HTML files.
(b) Deciding on patterns.
You should clearly describe all these decisions, the reasons behind your decisions and the properties of HTML files (URL, length, etc) in your detailed report.
Note that, the first character of the pattern, repetitions in the pattern, the length of the pattern, etc. may have significant effects on the running time of the algorithms. You should make clever enough decisions to reflect these effects as much as possible.

Step 2: Coding and Running: All algorithms should be implemented in any programming language and experiments should be performed for decided HTML files and patterns. Your codes should give the following outputs:
(a) Bad symbol table and good suffix table for each pattern,
(b) Output HTML file with highlighted pattern occurrences,
(c) Number of character comparisons and running time.
You should also run your code for the text and pattern given at the end of this document (Note 5) and provide the outputs to show that your code runs correctly.

Note that implementation details may affect the performance. Please provide any decisions on parsing the files, storing the values in the memory, and generating the highlighted file.

Step 3: Illustrating and Analyzing Results: This step includes:
(a) Illustrating the timing/complexity results in plots and/or tables,
(b) Comparing the performance of all the three algorithms for various HTML files and patterns.
(b) Providing detailed comments on your finding in these comparisons.