

Due 1:30PM Monday, December 6

Name \_\_\_\_\_

Student ID \_\_\_\_\_

**Instructions** Follow instructions *carefully*, failure to do so will result in points being deducted. It is highly recommended that you write your homework using  $\text{\LaTeX}$  or Word. Staple this cover page to the front of your assignment for easier grading. Be sure to submit a hardcopy of your problem answers and your program analysis in class. Clearly label each problem and submit the answers in order. Put your program analysis to the end. In addition, submit your homework including all the program files via the webhandin (<http://www.cse.unl.edu/~cse310/handin>). Late homework policy specified in course syllabus *will be strictly followed*. Be sure to show *sufficient work* to justify your answer(s). Numerical answers with no explanation will NOT be granted full points. If you are asked to prove something, you must give as formal, rigorous, and complete proof as possible. You are to work individually, and all work should be your own. The CSE academic dishonesty policy is in effect ([http://cse.unl.edu/undergrads/academic\\_integrity.php](http://cse.unl.edu/undergrads/academic_integrity.php)).

Problem	Page	Points	Score
7.3.1 (a) (b)	270	6	
7.3.2 (a) (b)	270	6	
7.3.7	271	6	
8.1.1	282	6	
8.2.6	292	8	
8.2.7	292	8	
8.4.5	304	10	
Program			
Correctness		30	
Style/Documentation		10	
Analysis		10	
Total		100	

# Program

Implement the Horspool's algorithm, the Boyer-Moore algorithm, and the brute-force algorithm of Section 3.2 that, for a given binary pattern and a binary text, outputs: (i) if a matching substring is found in the text or not; (ii) how many character comparisons are made by each of the three algorithms.

Your program must be written in C++, execute correctly on the CSE unix environment, and compile using the g++ compiler using a makefile. The makefile should compile your program into an executable called string\_matching. Your main.cpp program should take input from a file via the command line (example: prompt:> string\_matching inputfile) with the following structure in the input file (you may assume that an input file will be well structured so you dont have to error check the input).

Each line of the input file represents a pattern and a text. The first string in a line specifies the binary pattern and the second string gives the text.

An example of an input file is as follows:

```
01010 000000000
100 100111
```

It specifies two instances of string matching problem. The first instance is to search for the pattern 01010 in the text 000000000. The second instance is to search for 100 in 100111.

Proper output should look (something) like:

```
////////////////////////////////////
```

String-Matching1:

Horspools algorithm:	No Matching Substring	6 Character Comparisons
Boyer-Moore algorithm:	No Matching Substring	4 Character Comparisons
Brute-force algorithm:	No Matching Substring	10 Character Comparisons

String-Matching2:

Horspools algorithm:	Found Matching Substring	3 Character Comparisons
Boyer-Moore algorithm:	Found Matching Substring	3 Character Comparisons
Brute-force algorithm:	Found Matching Substring	3 Character Comparisons

```
////////////////////////////////////
```

Points will be awarded based on the following categories:

- **Correctness** – Your code must compile and execute as expected. You must include a **makefile** that will compile your code into an executable called **string\_matching**.
- **Style/Documentation** – Your code should be readable, properly indented and spaced with sufficient comments to explain. Good naming style should be followed.
- **Analysis** – For this homework, your analysis should include two parts. First, you need to explain your design of the algorithms. Second, discuss anything you think relevant, pitfalls that you faced, etc. Turn in your analysis as a hard copy with the rest of your assignment.