1. What is the main source of inefficiency in brute force string searching?

2. What is unusual about the comparison order in Horspool's string search algorithm?

3. If the text character that we compare to the last pattern character occurs nowhere in the pattern, how far right can we shift the pattern, and still be sure that we do not miss a match?

4. Explain the reasoning behind the rule for calculating t[c] in the Horspool shift table.

5. Show the values in the Horspool shift table for the following characters if the pattern is COCACOLA :

Character	Shift
C	
O	
A	
L	
M	

6. Why is the " – k" in the formula for Boyer-Moore bad-symbol shift?
$d_1 = \max\{t_1(c) - k, 1\}$, where $t_1(c)$ is the value from the Horspool shift table.

7. **Boyer-Moore Algorithm:** After successfully matching $0 < k < m$ characters, with a mismatch after k matches from the end of the pattern (the corresponding mismatched character in the text is c), the algorithm shifts the pattern right by
$$d = \max\{d_1, d_2\}$$
where $d_1 = \max\{t_1(c) - k, 1\}$ is the bad-symbol shift. $t_1(c)$ is the entry for c from the Horspool table.
$d_2(k)$ is the good-suffix shift

8. (4 points) With one or two other students, try to come up with rules for creating the good shift table for a pattern string of length m**. Input:** the pattern string. **Output:** a table of m-1 shift values. gs[k] is the amount that we can shift the pattern if the last k characters of the pattern match the text. [domain: k = 1..m-1]
   Example patterns to help you think about this: CABABA, AWOWWOW, WOWWOW, ABRACADABRA.

9. For each given string, fill in the good-suffix table from the Boyer-Moore algorithm. Once again, work with one or two other students.

1. banana

k	shift
1	
2	
3	
4	
5	

2. wowwow

k	shift
1	
2	
3	
4	
5	

3. abcdcbcabcabc

k	shift
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	