CS340 Winter 1996
Ming Li

Assignment 4, Due Date: March 22

1. (1 mark) Give a complete analysis of the average case time complexity of the QuickSelect algorithm given in class.

2. (Programming, 9 marks) Implement a generalized version of Knuth-Morris-Pratt linear time pattern-matching algorithm. Your program should accept two character strings from standard input: the text string on the first line, and the pattern string on the second. The program outputs a list of integers which indicate *all* positions where the pattern occurs in the text. If the pattern does not appear in the text, then it outputs -1. For example, on input 'alfalfa' and 'alfa', your program should output 1 and 4 to indicate that the first occurrence of 'alfa' is at position 1 and the second occurrence of 'alfa' is at position 4; and on input 'alfalfa' and 'alpha', your program should output -1.

   Your program *must run in linear time* in terms of the lengths of the text and the pattern. As a part of this question, you should prove this fact and show your program will correctly report all occurrences of the pattern. Trivial implementations (of the straightforward $O(|p||t|)$ time algorithm) will not receive any marks.

   To test your program, run your program on the following data.

text	pattern
alfalfa	alfa
alfalfa	alpha
couscous	cous
beriberi	beri
undergrounder	under
toronto	to
abababababab	aba
to test your program, run your program on the following data	program
to test your program, run your program on the following data	progrm

   You should also do some testing with your own data with longer texts and patterns.

   Hand-in: For this assignment, submit your source code and a make file called "Makefile" via the "submit" program. The executable produced should be called "match". Also, hand in a hard copy of your source code to the assignment boxes, along with the testing described above.

   Marking: 2 marks will be for the proof of linear time correctness; 3 for the code (correctness of the algorithm, modularity, etc.); and 4 for functionality (2 for the tests given above, 1 for your own testing on longer data, and 1 for the auto-testing that we do on your electronically submitted code to verify that the submitted code compiles and runs.)