Programming 4

Comp 221

Susan Fox

Guidelines

Goals and intentions: Being able to convert pseudocode into actual, running code is an important skill. Thus these assignments ask you to practice just that. In order for the practice to be valuable, you must write the code yourself. Finding a complete solution online and getting it to run is not the point, and is not acceptable. While you may use built-in or provided data structures for many of these tasks, you should **not** seek out online solutions.

These assignments are separated from the regular homework for two reasons. First of all, homework is due weekly, and these assignments have a longer time frame. Secondly, while homework should be individual work, I encourage (even implore) you to work in pairs on these programming assignments.

Programming languages: You may use any normal programming language of your choice, so long as I have access to a compiler or interpreter with which to run it. The standard languages include Python, Matlab, Java, C, and C++. I am providing solid starter code in Python and Java. If you find bugs, please post to Piazza about them so I know to fix them, and so that everyone knows that they are there.

What to hand in: You must hand in your code and all required supporting files. You must also write a report and hand that in as well. I will be both reading your code and running your code. Therefore, your programs must be clearly written, well-organized, and using good programming style. You must include good comments, meaningful variable and function names, etc. In addition to your code files, you must hand in a brief report with a section for each program. For each program, tell me where to find the source code for each program, and how to compile and run your code (what version of Python, for instance, which file contains the main program, etc.). Tell me about your program: does it work correctly on all inputs, how did you test it? In addition, answer any questions about the problem or program that are given in the assignment questions.

Put all your work in one folder. Title the folder with all teammates' first names and the assignment number. Create a zip archive of the folder, and upload that to Moodle.

Programming Problems

Choose TWO of the following three problems to implement.

Heap Construction (20 pts))

Implement the bottom-up heap building algorithm, that operates on an array of numbers and treats the array as a heap. You do not need to build some kind of heap data structure, just move the data values around in the array in a way that is consistent with a heap. I recommend that you come up with formulas for determining parent, leftChild, and rightChild using zero-based indexing (for simplicity), and define helper functions to do that calculation for you.

Test your algorithm and time its performance. Does its actual performance match the theoretical efficiency of O(n)? Discuss your results.

Heap Sort (20 pts)

Implement the heap sort algorithm, operating on an array of numbers and treating the array as a heap. You could use the previous problem as a helper for this one.

Test your algorithm: does it work? How does it handle equal values? Is it stable? Time it, and see if its performance matches the O(n log n) predicted efficiency.

Horspool's Algorithm (20 pts)

Implement the brute-force string-matching algorithm from chapter 3, and Horspool's Algorithm for string matching. Your program should keep track of the number of comparisons for each algorithm. For Horspool's, also have your program print out the shift table (just the characters with a non-default shift).

Test your program against the brute-force one, counting the number of comparisons done. How does Horspool's algorithm compare? In the worst case, can it be as bad as brute-force?