Programming 3

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 ONE of the following two problems to implement.

Closest Pair Divide-and-Conquer (20 pts))

Implement the divide-and-conquer Closest-Pair solution from the book, working from the pseudocode that I posted online. In your report, talk about the process of implementing this algorithm and issues involved in converting from high-level pseudocode to implementation. Note any special cases that you needed to handle (two points at the same location is one special case, can you find others?). Thoroughly test your program, and describe your testing method in your report (you may provide me with a testing program/file as well, if you like). Do efficiency testing as well: time the performance of the program on different sets and different sizes. Does the performance match the analysis of the algorithm?

AVL Tree Insertion (20 pts)

Look at question 6 from Section 6.3. You are to implement the insertion algorithm for AVL trees. I have provided you with starter code that implements a simple BST. You are to modify this code (porting it to your preferred language if you don't want to use Python or Java) so that it keeps track of the balance factor for each node, and performs the rotations correctly when insertions happen. You do NOT need to implement AVL deletion.

I have provided three implementations of a BST data type: BSTADT.py, BSTClass.py, and BST.java. The first implementation is for those who are not comfortable with object-oriented Python programming yet. It is a function-based implementation of the BST ADT in Python. The second implementation uses object-oriented Python, giving a BST class. And the third implementation is similar, defining a BST class in Java. Each program includes a main program that tests the data types.

Create a series of tests of your implementation to demonstrate that your function always performs rotations correctly. Be sure to test all four kinds of rotations. Use the printTree operation I've provided to display the tree before and after an insertion.