Engineering and Applied Science Programs for Professionals
Whiting School of Engineering
Johns Hopkins University
605.421 Foundations of Algorithms
Programming Assignment Guidelines

Typically, the programming assignments will be designed where you will implement algorithms and compare their performance on data sets of your choosing. For these assignments, you are required to submit source code, trace runs that demonstrate proper functioning of the code (including inputs, outputs, and intermediate results demonstrating proper functioning), a brief description of your conclusions and a ReadMe file. These programs typically complement individual problems in the homework assignment for the modules in which it is assigned. To facilitate the grading of programming assignments, the following additional requirements must be followed:

- Some versions of the CLRS textbook come with a CD containing Java code for all of the algorithms in the text. You are allowed to use the code to complement your programming assignment. You must submit the individual files you use from the CLRS code not the entire set.
- All source code and supporting materials (described below) shall be submitted as part of the assignment when your programming assignment is submitted.
  - Eclipse IDE is required to create a Java Project for each Programming Assignment.
  - If other IDEs or command line is used specific instructions must be provided in the ReadMe.txt.
  - A main file that runs within the Java project you create must be provided.
  - All supporting files for the main class should be included in the submission.
  - Code must be documented, when commenting your code make sure to provide a brief list of any modification at the top of the code. Within your code you can provide specific detail. It is common practice to use JavaDoc-type comments at the start of a method that explains what the method is doing. In addition, the code should use method and function names that are "self-documenting." For example, use "index" rather than "i" or "data[ ]" rather than "a[ ]". Refer to Javadoc Tool for information on documenting your source code in a manner suitable for use with javadoc.
  - ReadMe.txt must include a list of the files in the project, specify the compiler (JDK version) as well as the IDE (Eclipse), specific compiling instructions, specify specific instructions to run your code.
- The source code and supporting materials shall be zipped into an archive file where the name of that archive file will be <lastname><assignment#>.zip or <lastname><assignment#>.tar.gz (or whatever is appropriate to your compression software). The "#" should be replaced with the number of the assignment.
- Packing your work for grading - although you will create a Java project, it is very important that you produce a directory containing only the source code you wrote or modified. Otherwise your software is contained somewhere in a deeply nested hierarchy and is quite difficult to find and grade. I would appreciate anything you can do to clearly delineate your work on the assignment for efficient and effective assessment
- When writing your programs, you need to instrument them in a way that the code produces an output file/results of the important points of its execution. Each student is free to include a switch in the code to permit the output to be turned on or off, but you are required to submit a file containing the ouput run(s) with your program. This could be included in your ReadMe file.

- Trace runs - Please refer to Tracing and Logging and Tracing Tab in Eclipse for mechanical references related to traces. The objective of a trace is to demonstrate the proper functioning of your code. This is usually demonstrated with one or more data sets designed to specifically exercise the program. One quick source of test types information is Types of Software Testing.
- Sample test results are required of all programming assignments. The results of each test run should appear in a file(s) (preferred ASCII) with two sections:
  - inputs
  - outputs
- Test your implementation to verify the predicted complexity bound - You must not use CPU time or clock time to record the efficiency of your code and algorithms as these metrics are not fully under your control. You must instead identify key actions in your code that indicate the amount of work being done for a particular input length. This might be the number of times key loops or branches are executed or the number of times key functions and procedures are called. You will need to test your program with input list of length such as n=100, 10000, 500000, 1000000. You will probably have to randomly generate integers (most algorithms work upon integers in this course) for test case inputs. You will need to analyze the results of your tests of varying n sizes and determine if your results grow at the predicted complexity rate. This analysis will be in your brief description of your conclusions.