Coursework commentaries 2015–16

CO2226 Software engineering, algorithm design and analysis

Coursework assignment 2

General remarks

This report provides a general commentary on the questions and the performance of students for the second coursework assignment of the CO2226 course. A comment on the expected answer, in generic terms, is provided as well as comments on student performance and common mistakes made in answering the questions.

This coursework assignment was a programming exercise where students were expected to answer five questions. The questions were based on the implementation of Dijkstra's algorithm for the shortest path problem and fragments of code were provided. You were expected to modify them and add your own code to answer the questions. Some questions were based on earlier answers, with two main criteria (apart from, obviously, correctness) namely, the speed of execution and re-usability of data structure components so that the same calculations did not have to be made from scratch. Full marks for each question depended on whether the result produced by the student was correct.

Below is a list of the **common mistakes** identified in the submissions; please note that these mistakes were penalised with marks deducted from the total marks allocated for each question.

- When IDEs (e.g. Eclipse, NetBeans, etc.) were used for writing the code, the IDE inserted a package statement at the beginning of the code. This line had to be removed; otherwise, an error would be generated after compiling.
- The use of IDEs also results in a structured way of writing code (e.g. code files in a directory called src, data in a directory called 'data', libraries in a directory called 'lib' and so on), but this broke the assumption when running the file that all data files about stations, lines and edges should be in the same directory as the main file. This is easy to fix as all that had to be done was to remove the prefix 'src' so that all files could be found in the same directory.
- Some students made use of third party libraries, mainly for the reading of data from the source files and their presentation – instructions were included in the assignment about compiling the source files with the library files included in the classpath to enable the code to work. In this case, there were no error messages when compiling the file but the coursework specification clearly states that no additional resources should be used apart from the code given in the coursework specification document. Nevertheless, because in this case the additional code was not used for any processing on the graph aspect, these submissions did not receive zero points but the marks were deducted.

- Another common mistake was forgetting that inner classes were required (in this case the class Graph). If this class was omitted, the program would not compile since the constructor for the Graph class is used in the program. Since the program could not run, no marks were given.

In general, you should make sure that your code compiles without errors. Pay close attention to any error messages you get, understand why you are getting them and what you should do to get rid of them. Also, think very carefully about what data structures you should be using and why. You should put effort into understanding what information is required to answer the questions, and consider how you can reuse this information. A program that re-computes information that has already been computed is slow, ineffective, harder to debug and counterintuitive.