Coursework commentaries 2016–17

CO3325 Data compression – Coursework assignments 1

General remarks

This coursework requires building a prototype Java program to demonstrate how the Adaptive Huffman Coding algorithms work. The input and output for the program are outlined, with an example suggesting what to display in interim steps. Hints are given to indicate how to go about the coursework (e.g. design before implementation, a modular approach and linking various components by the menu.java provided in the Appendix). The submission requirements specify a report document and running Java program codes, and includes a general marking scheme.

A good approach is to simply follow the given instructions and guidelines, for example doing exercises to understand the relevant concepts before attempting the assignment tasks, reviewing the software engineering approach (see subject guide Vol. 2) to complete a software development journey in a professional way. The milestone stages include, for example:

1. working on the given problem specifications → designing rough solutions in block diagrams →
2. designing fine solutions in class diagrams → finalising the design with general analysis → class diagrams or pseudocodes →
3. testing →
4. evaluation of your software.

Comments on specific questions

Most students did very well in their coursework, using precisely the recommended approach. They followed the requirements and guidelines carefully, and submitted both the report file (in .pdf) and the programming Java source codes (in .java) as required. They demonstrated their software and reported useful learning experiences and interesting discoveries. These students not only achieved good grades, but also benefited from useful preparation for the examination. Disappointing coursework included partial submissions, missing documents, incomplete work, poor understanding of the assignment tasks, little sense of problem solving, flawed design or no design at all, over-reliance on others' work (typically adopted from the internet) and superficial or no discussion.

Success in coursework often reflects the effort expended. Aim to present a complete piece of work, starting with careful design; include comprehensive details serving clear purposes, choose appropriate data structures and explain decisions enabling efficient approaches to problem solving. Reports should be clearly structured and well-written, with insightful discussion on technical issues, informed arguments and reflection on personal experience in tackling the assignment. The programming style should be neat (indentations etc.) with sufficient comments. Screenshots should be selected to best highlight the program execution.

Coursework commentaries 2016–17

CO3325 Data compression – Coursework assignments 2

General remarks

This coursework assignment requires exploration of JPEG preprocessors. Tasks are diverse and divided explicitly into three parts. Part I requires the development of prototype Java programs for a preprocessor for grey-scale images. The program specifications are further divided into three sections. Examples are given to indicate program interfaces. Part II requires a plug-in of the main compressor using the Adaptive Huffman program (or another lossless compressor if unavailable) from the previous coursework. Part III requires mini-experiments on the impact of the preprocessors in compression.

A straightforward approach to this coursework is again to follow the instructions given in the assignment specification. For the programming tasks in Part I, the best approach is to follow the software engineering approach suggested for Coursework 1 (above). For the tasks in Part II and Part III, a feasible plan is necessary to decide not only on the solutions but also on how to present your work, based on the best understanding of the requirements.

Comments on specific questions

Most students did a good job for this coursework. Many enjoyed attempting the coursework assignment and reported interesting learning experiences with enthusiasm. High-quality submissions often involved elegant designs, and demonstrated a strong grasp of critical issues. The work presented was typically precise, coherent, comprehensive and imaginative, including insightful discussions. In contrast, weak coursework submissions were often incomplete or superficial, lacking in structure and presenting somewhat vague ideas, with limited design and coding. Such work typically lacks discussion. The weakest submissions showed vague understanding of key issues, and minimal problem-solving skills; poor time management appears to be a factor here.

It is always a great pleasure for the examiners to observe good practices. For example, some students realised that their programs could easily be extended to handle true colour images, so they offered additional functionality. Other students discussed topics well – sometimes even where they had not managed to complete all the required tasks.

It is interesting to observe that some students did not follow the advice about learning the basics and attempting the exercises before attempting the coursework. Instead, their starting point was to search for an existing solution, without properly studying the problem. This tends to lead to difficulties (e.g. in presenting design choices, and discussion of issues) and of course the learning experience is less rewarding.

Some students were reluctant to make their own decisions, preferring merely to code according to given specifications, rather than making their own designs. Understandably, making a design can be much more challenging than coding based on an existing algorithm, as designs often require more analysis, many decision choices, and development from broad ideas. Of course, in in many real-world situations, customers have only vague ideas about what they want from the software system. An important initial step in any software

development is to specify the requirements and identify the algorithmic problems. Reasonable assumptions may be required to simplify computational problems. This assignment mimics this to prepare you for the real-world problem solving.

The final task requires further consideration and planning as it is research-based. Hypotheses should be proposed before experiments, and appropriate measures should be used for comparisons. Conclusions should be drawn to summarise the findings for or against the previously stated hypotheses.

Good coursework focuses on design before implementation. Some students apparently only realised the importance of design after completion of the whole coursework. Of course, it is never too late to learn from experience. Those who reflected on both their findings and their experience will be able to do better in their next software development project.