EE273/985 - Project Guidelines, 2016/17

David Harle
February 2017

1 Introduction

The last weeks of this course are to be spent working on a software project. Students are required to work in pairs to develop the C++ program required to perform a particular task, demonstrate it working, describe the code used and submit a full project report. Failure to participate fully in the Project will result in failure to pass the course.

Students should spend the remaining lab sessions working on projects. Attendance at lab session is not optional – it is monitored and students will be required to have logbooks countersigned at each lab.

2 Project conduct

1. Students have been put into project pairs; the full list will be posted on MyPlace. Students will work on the project within the given pairing – it will not be possible to change the given pairing. Each pair has been allocated a particular project title/descriptor.

2. Each pair should work on a draft user requirements specification and project plan to be submitted to staff at the end of the 1st project lab session. This document will be extended and revised and formally submitted at a later date. The motivation behind the first version is ensure a fast start on the project and give students time to reflect on what is to be achieved in the project. Committing to a first pass of requirements and plan will focus attention.
3. Each pair should submit a single page 'user requirements specification' of their project and a project plan to the EEE Resource Centre (MyPlace) by **18.00 on Tuesday 21st of February 2017** (i.e. at the end of week 6's lab session). Include the names of the team members and the title of the project. Late submission will incur a penalty of 5% of the marks available for the project.
   - The project 'user requirements specification' (URS) should describe in, general terms, features of the final application.
   - The project plan should list and describe the main project tasks necessary to the delivery of the specified program and a written report. These will be important to you when you try to organise your work and how you split the tasks between the team members, so write them seriously. The plan must be included as a permanent item in your logbook and referred to regularly (and revised accordingly) as the group reviews progress.
   - The URS and plan should be printed out and fixed into each group member's logbook and be able for view by TA and staff on request.
   - It is likely that the functional specification and plan may be revised as the Functional and test specification are developed (see next) over the next week. This is expected and it is appropriate include a revised URS and plan in your logbook. (Retain your 1st version in your logbook.)
4. Each group should submit a combined Functional & Test specification. This should be submitted MyPlace no later than **12.00 on Tuesday February 28th 2017**, i.e. week 7.

Include the names of the team members and the title of the project. Late submission will incur a penalty of 5% of the marks available for the project.

- The Functional specification considers in detail the function of the application in light of the general design stage that has taken place over the previous 7 days.
- The 'test specification' details how the functionality of the program will finally be tested and how it will be ensured that it satisfies the 'functional specification'.
- It is expected that a revised plan will be required that describes explicitly the roles of each group member and group member's individual responsibilities and along with interim deliverables and dates.
- As well as submitting the Functional & Test specification to myplace, each group member must include a copy of the submitted document in their logbook along with any revised plan.

5. Students must be ready to give a demonstration of their program to a member of staff or Teaching Assistant (TA) on a date to be communicated when the University publishes its examination timetable.
    - It is expected that the project demonstration will take place within the examination diet at a date scheduled by the University.
    - Failure to give a demonstration (without certified reasons) WILL be considered as a non submission and awarded 0 marks.
6. Arrangements and schedule for demonstrations will be communicated to students prior to the week 11. Students are expected to make themselves available for the entirety of the session.
7. Please note, the signing of a receipt and program demonstration does not, by itself, imply that a project is satisfactory and that a pass has been awarded. Students will be informed about course results via the normal channels after the June examination board.
8. Project reports should be handed to a TA or member of staff **at the start of the demonstration** in the demonstration session. An e-copy of report plus code must be submitted via myplace **BEFORE** the day of the demonstration. (The date will be clearly marked on myplace and confirmed in due course.)
    - Any late submissions of project reports will incur a penalty of 10% of the maximum project mark for each day late, including Saturdays and Sundays. Late submissions should be handed in to the EEE Resource Centre and a receipt obtained.
    - It is expected that all groups will complete their project prior to the start of the examination period. It is understood that students have to prepare for examinations taking place from 18th April onwards and that groups should organise their time accordingly. Preparing for other class examinations is NOT acceptable grounds for non-completion of project, late submission or substandard submissions - report or code.
9. Students must attend ALL appropriate lab sessions.
    - A register is taken. As this a group project, it is important that all group members participate and contribute. Students cannot expect other group members to do their work for them and then to gain credit for the efforts of others.
    - Students are expected to indicate to staff if there are issues with regards to progress and participation of group members. Running the project implies a degree of personal responsibility from all group members.

- Students who miss one lab session will be deducted 20% from their individual project mark while missing TWO lab sessions will be NQ'd from the cmodule. Students who do not complete
- It is likely that you will need to do work additional to that spent in the lab sessions. Time spent outside the timetabled lab sessions should not be viewed as a substitute for work inside the sessions. The lab sessions will be your only chance to get help.

10. The project is worth 50% of the final marks of the class and thus there is an expectation of a commitment around 60 hours for the project from each student.
11. The information required for any particular project is not necessarily contained in the lecture notes and perhaps will not have been covered during laboratory sessions.
    - Textbooks, web resources, data sheets and staff are the appropriate sources for any additional information. Credit will be given for the quality of a student's research but is no substitute for a working program. Unacknowledged use of other people's resources is not permitted.
    - It is encouraged that students do consider the use of 3rd part libraries or APIs in their project – either to provide specific functionality specific to the project task or more general functions such as GUI, data graphing, or stats or mathematical features. Any decision about such use must be taken early. Please consult with TA staff about the applicability and practicality of using such libraries.
12. While you will receive credit for identifying and using publicly available code or information that enhances the meeting of your objectives, provided you have the right to use it and the source is properly acknowledged, most of your marks will be awarded for code you have written yourself.
    - Even if you manage to find and use code or libraries that do some of what your project requires and can be legally downloaded, you should still write a substantial amount of your own code and you should be able to explain the form and operation of any code that you adopt from elsewhere.
13. This project accounts for 50% of your overall mark. Please note it is a requirement of the course that you must submit a satisfactory project to pass the course and receive the credit.

3 Project Report

Students must submit a typed project report – typical length around 5000 words plus diagrams and code listings. This should be bound in a soft plastic A4 folder with a transparent front. It should be clearly marked with the students' names, the project title and the class code and should be submitted along with a completed standard EEE department coursework cover sheet (available from the Resource Centre). The report must contain the following elements:

1. A table of contents
2. An introduction to the report
3. An outline description of the objective of the project and the 'user requirements specification' of the program.
    - This should describe the objective of the project in general terms, i.e. to a reader with no prior knowledge of what you are trying to achieve. The user requirements specification should be based on that submitted in week 7. Any changes to the specification from that submitted in week 7 must be explained.
4. The 'functional specification'

- A description of the functionality need to meet the user requirements.

5. The program design
   - A detailed description of the program design, how it will meet the 'user requirements specification', the classes used, their data members and functions and how the classes are used and relate to each other. The program solution should be object-oriented and modular. UML could be used to illustrate the relationship between classes.
6. Results
   - This should describe and summarise testing of the program (in accordance with the test specification given in the appendix of the report) and show its outputs for different inputs
7. Discussion
   - A critical reflection on the program, the test results and whether the project was successful in meeting the objectives, how robust the software is and what its limits of use are.
8. Further work
   - How the program might be improved.
9. References
   - List which texts (books, articles, webpages, etc.) or other resources were used to inform the work, the names of their authors, when and where they were published.
10. An appendix containing
    - a user guide;
    - the 'test specification' – a defined set of actions designed to prove that the program works correctly;
    - fully commented code listings. Each module of code should start with comments detailing the name of the project that it forms part of, the name of the module, the date on which it was last updated, a brief description of the last update, and the name of the author of the module. Further comments in the code should enable a third party – albeit one at least a little bit familiar with C++ – to understand the code and maintain it.

In the report, use a formal writing style and do not use the active voice (I, we, you etc.) in your report. Use the passive voice, e.g. 'the software was written' rather than 'I wrote the software'.

Be concise in explanations/descriptions. Longer reports are not necessarily better.

4 Plagiarism

Discussion and collaboration with others is a vital part of project work. (One of the oft-cited advantages of an object-oriented approach to programming is that it enables collaboration on a single project among many developers, and a modular design). However, you must give credit to any sources used from outside your project team.

Failing to credit the precise extent of any material not produced by your team is plagiarism and is treated very seriously by the department and the university.

If in any doubt, ask a member of staff.

5 Marking scheme

The project is worth up to 50% of the final mark for EE273/985. Final assessment will consider the following aspects. – note that final mark is not just additive but is multiplicative.

- A functioning program that meets the specification;
- Use of advanced programming features;
- The written project report;
- Demonstration of the final program.

In the demonstration, each member of the project team must actively participate and describe the function of different aspects of the code. Each half of the marks will be awarded based on the performance in the demonstration of one team member. **It is therefore extremely important that <u>ALL</u> team members fully understand the program**, including any modules that were designed and written by their colleague, and contribute to the project. A student's final mark will take into account of individual contributions to the project; based upon performance in demonstration, documentation in reports and logbook, attendance at lab sessions.

The following are among the advanced features for which credit will be given for successful and appropriate use:

- class inheritance
- separation of interfaces from implementation
- operator overloading
- pointers to build associations
- references
- containers (vector, lists or both)

6 General remarks on a programming assignment

Successful programs are generally those that have a sensible structure and which make use of features available in the programming language that allow the program to be concise. These include library functions for input, output and various mathematical procedures but also standard programming ideas such as arrays, loops, e.g. do or for, and branching using if. These can avoid repetition and improve reliability of the code. Many of these features have been covered in the lecture course and tutorials.

Object orientation offers some particular features to a program that should allow it to be more easily maintainable, better designed and expandable. C++ is a language that permits object oriented programming. To take advantage of the features of object orientation requires some thinking and design before any coding is started. For example:

- what are the 'things', i.e. objects, that your program will be concerned with, e.g. something representing a person, or a robot or a bank account or an equation?
- what are the relevant attributes of the objects that your program will deal with? e.g. for a student it might be name, gender, degree course, modules taken, results gained. (What are the characteristics of the objects that are described by *values*?)
- what are the things that you want to do with your objects or that you want your objects to be able to do?. (What are the characteristics of the objects that are described by *actions* or *functions*?)

You should start your work on your assignment by thinking about what exactly you want your program to do and how you want to do it. The briefs given below are just that – brief. They outline the end results that are required but, while they also make some suggestions, do not tell you exactly how to do it. That is up to you to decide. You will need to interpret the outlines given below to write a user requirements specification (URS); show your draft URS to a teaching assistant at a lab session and get their comments before submitting it.

Start the software development by writing down outline definitions of the classes you want to use, the data members (‘attributes’) and the member functions (‘methods’) and what the functions will do.

For data, do you want to use any sets or series of data? Will you have multiple instances of objects? Will each item have similar actions carried out on them, in which case could the items best be captured in vector, lists or arrays so that these repeated actions can be implemented inside loops? (This saves on repetition of code and cuts down the chance of error. Anything that is ‘iterative’ is clearly best implemented in a loop). Then, think about the processes that the functions will implement and perhaps sketch a flow chart or write down some ‘pseudo-code’ describing the logical steps and the condition to be tested when the action of the program might branch in two or more ways. When you have done this and are happy with it, you will be ready to start coding what you have designed.

When you write your code, make sure that you lay it out neatly (use clear and consistent indentation!) and include comments explaining what is going on – not only will it help anyone assessing the program, it will help you remember what you were trying to do when you come to debugging. Include code that will deal gracefully with possible erroneous use by the user.

In your project, think about how you will test that your program works correctly, and how you will demonstrate the full range of its capabilities. For example, you could test your code by entering input data for which you already know what the output should be: does the program successfully produce the expected output? Then enter data or inputs that result in unknown outputs or non-function code.

Make full use not only of the course material but also books, and guides that are available on the web. Be mindful, however, of the university’s rules on plagiarism – the majority of the code in your assignment should be your own (with any other code fully acknowledged in code listing and final report).

7 Project outline requirements

The project descriptions – listed in a separate document (available on MyPlace) – are not full specifications. Some are more comprehensive than others. As the system designers, it is up to you to develop as complete a system specification as possible. You should speak to staff and TAs to elicit advice and direction and also canvas views of the potential users of the systems. Students are also required to do some further research regarding the project topic itself and also additional libraries and operating systems.

8 Summary of milestones

Milestone	Date and time
Title of project and names of team members	13.00 Tuesday 21st Feb (week 6)
Provisional user requirements & project plan	18.00 Tuesday 21st Feb (week 6)
Functional specification and Test specification	12.00 Tuesday 28th Feb (week 7)
Demonstration and submission of project report	after Tuesday 18th April (week 12)