EEE101 C Programming and Software Engineering 1 – ASSESSMENT 4

Assessment Number	4
Contribution to Overall Marks	35%
Issue Date	13/11/2017
Submission Deadline	15/12/2017 at 17:00 (5pm)

Assessment Objective

This assessment aims at evaluating students' ability to develop a significant software solution to a real-world problem by working as a member of a team. Your team will be given a vague specification and is expected to deliver a software product in the C programming language, which meets the specifications before the due date. This size and type of the project is suitable for development in modular format and so teams are encouraged to devise program structures that allow various parts of the code to be developed independently by each team member. Being a team player means you are expected not only to apply the knowledge gained during the lectures, laboratory classes and assignments to specify, design, implement, test and document your own code, but also to cooperate with your teammates so that the whole project will be delivered on time with good quality.

Grouping

There are 228 students enrolled in this module, and you will be divided into groups consisting of 5 students (and 2 groups of 4 students). Groups will be formed in two stages as follows: Firstly, students will be given the option to choose their own group members. Students failing to form a group will then be randomly assigned to a group. Randomly formed groups will contain students with a range of ability based on their performance in previous assignments. Each group will then be randomly assigned one of 5 projects.

Final Deliverables

Each group should submit the following:

1. A report (a single MS-Word or PDF file), provides the following details based on the Software Development Process applied throughout this semester:
    a) Problem Statement (Specification: formulate the problem generally.
    b) Analysis: interpret the vague software requirements provided in each design brief and determine a very clear specification for your software design.
    c) Design: explain how your program is structured, what each functional block does and if possible why you have chosen this method.
    d) Implementation: the C source codes (have to be included in the report).
    e) Testing: explain how the program has been tested and verified.

2. All C source code (.c files) and the final executable demonstration file (.exe). The source code must be commented appropriately for implementation.
3. A simple program manual (MS-WORD, or PDF), describing your programs basic functionality (how it works), known bugs, and functionality status.
4. A personal contribution statement (1 page) from each member of the group that describes (i) how he or she contributed to the group, (ii) what are the main technical difficulties he or she encountered in the project.

NOTE: Each group member must submit a personal contribution statement. This may lead to different marks for different members of the same group. If necessary, the module leader may call for a short oral test for certain groups.

Submission Procedure

All of the above-mentioned files/deliverables (report, source code, executable, manual, personal contribution (1 per person)) must be zipped into a single file (RAR or ZIP). Then, the coordinator of each group must submit this single file on ICE using his/her account.

NOTE: Each group should only submit **ONE** copy on ICE. Make sure your report has a title page and ensure **ALL** group members names are on it.

Marking Scheme

This assessment requires the routine of code development using the software development process. The general marking scheme is shown as follows:

Documentation	**(55%)**
Overall Quality of Report	10%
Specifications	10%
Analysis	10%
Algorithm Design	10%
Testing	10%
User Manual	5%
Coding	**(45%)**
Implementation/coding style	35%
Robustness	10%

A detailed marking scheme is attached (filename: 2017_Assignment 6 Marking Scheme.pdf)

General Guidelines

The project descriptions are deliberately given in the form of simple customer specifications, which (as in the real world) are incomplete and often ambiguous, rather than a set of exact functional specifications. The group members should work methodically together (as the developers in a real world software project would) to:

1. Analyze and formalize the customer specifications (at this stage, the various design choices and the software features can be subject to the group's creativity).
2. Design and decompose the functional and programmatic aspects of the problem and allocate constituent tasks to each group member. You are expected to use a top-down design which can then be modularized so that the tasks for each member can be clearly determined. Designs which mimic object oriented programming (by using abstract data types) are encouraged although not required.
3. Implement the product with frequent meetings to report progress and decisions to each other and re-evaluate the agreed courses of action.
4. Implement test procedures, debug and correct the program. Each program module should be independently testable. Testing of each module and the program as a whole should be performed.
5. Finalize the deliverables.

The specifications are only basic and most of the design choices should be made in your group meetings. The systems described within the different projects have a variety of different features and the disambiguation of the customer specifications can be based on the student's logic and real life experience.

Assessment will be based on whether the product/program offers reasonable functionality and features (for the group size, allocated time and project difficulty), its design quality, flexibility, robustness, software bugs and other stated deliverables.

If the group cannot implement all of the system features mentioned, it is better to have a few features fully working without run-time crashes than none of the required features working properly due to bugs or disrupting ripple effects between modules in the project. However, the corresponding marks deduction will be applied depending on the missing features.

Project A: Library Information System

Overall description:

The university library needs a new electronic rental system and your team is employed to build it.

Customer specifications:

The implemented system should be able to handle the basic operations of a library including the following features:

- Catalogue the library books; this should be stored in an indexed order (e.g. by author name or shelf-mark).
- The information about each book title should include: author(s), title, ISBN, subject, loan type (normal, short loan, no-take-out), shelf-mark, loan status, number of copies, etc.
- Provide search functionality so that any user can find a book

System Users

The system should be able to provide functionality for different users listed below:

- **Administrator** who should be able to:
  - Add and edit book information including mark a book as lost or damaged
  - Register new library users as staff or student, each with varying privileges (e.g. staff can keep books longer) including the user's personal details.
  - Ability to extend the loan period for a borrower's existing loan
  - Print a list of available and borrowed books
- **Borrower (Staff or Student)** who should be able to:
  - Ability to borrow books
  - Ability to edit their personal details
  - Ability to renew their current loans for a pre-set number of times.

Project B: Hotel Management Information System

Overall description:

Your team is employed by the conference centre hotel to implement a software system responsible for the overall management of room booking and customer records.

Customer specifications:

The implemented hotel systems should be able to provide facilities to:

- Manage bookings for 100 rooms (20 per floor) and four classes of room (**, ***, **** and VIP). Each room is assigned a single price class.
- Manage customer accounts
- Offer hotel business statistics e.g. numbers of VIP customers, average numbers of hotel guests.

System Users

The system should be able to provide functionality for different users listed below:

- **Manager** who will be able to:
  - Set/amend classes for each room and the price per class. Each room should have a single price.
  - Provide a fixed discount on the regular price of rooms for special circumstances, e.g. group bookings.
  - Manage a customer database (add/edit/remove customers)
  - View hotel business statistics.
- **Booking operator** who will be able to:
  - Register a booking (by recording a customer's name, address, telephone number and hotel member card no.). Customers without a hotel membership card, cannot book a VIP room.
  - A search facility should be provided for room availability and dates. Additionally, the operator can book one or more rooms to a registered customer and at a regular or discounted price.
- **Check-in/out operator** who will be able to:
  - Record the arrival of a customer in the system
  - Edit booking details e.g. period of stay or room.
  - Check-out customers by calculating charges or even change the payment from the regular to the discounted price if the customer was dissatisfied.

Project C: Bank Information System

Overall description:

Your team is employed by a bank to implement a system to manage the banking affairs.

Customer specifications:

The implementation of the banking system should be able to provide facilities to:

- Register a new customer and store details such as name, address, telephone number, 6-digit personal identification number (PIN) number (security code for using the card) as well as some extra information (e.g., type of identification presented for joining the bank), etc.
- Store and manage customer account information. Customers can have one or more accounts and each account is uniquely identifiable by an account number generated by the system.
- Collect and display bank statistical information e.g. number of customers, number of accounts.
- Allow the following banking activities:
  - Display current account balance
  - Record and display all banking activity
  - Allow withdrawals from an account.
  - Register a deposit

System Users

The system should be able to provide functionality for different users listed below:

- **Manager** who will be able to:
  - Access customer account information.
  - View banking statistics (as described above).
- **Bank clerk** who will be able to:
  - Add/delete/edit accounts for an existing customer.
  - Make deposits to a customer's account.
- **Customer** who will be able to:
  - Access their account information and perform the banking activities (as described above) **except** the deposit of money.
  - All banking activities require the customer to enter their PIN number.

Project D: DVD Rental System

Overall description:

Your team is employed by a DVD rental company to implement a software product to store their movie database and handle customer rentals.

Customer specifications:

The implemented video-rental system should be able to provide facilities to:

- Store movie titles, number of copies, title information (e.g., directors and actors), age limits (e.g. films for children or films not suitable for children) and genre (i.e. type of the film, horror, action, drama etc.).
- Store customers' information, identifiable by a unique pass number. Customer information should include name, age, telephone, address, pending charges, rental history, any rental restrictions, etc.
- Provide facilities for search by title, actor and/or movie director.

System Users

The system should be able to provide the functionality for different users listed below:

- **Branch manager** who will be able to:
  - Add new titles and rental copies as well as edit/delete them.
  - Specify rental duration and set/alter charges
  - Add/remove/edit customers information (see above)
  - View current stock status by listing all titles and the number of copies available/on loan.
- **Rental desk worker** who will be able to:
  - Allow existing customers can rent available titles or return current loans
  - Pay charges and penalties.

Project E: Parking Management System

Overall description:

Your team is employed for the implementation of a parking payment system in a university car-park. The parking space consists of:

- A 15x15x10 multi-story car-park, where each of the 10 floors is a 15x15 rectangular grid of car parking spaces;
- A 8x8 rectangular grid of e-bike parking spaces each with an electrical recharge plug provided;
- An 20x20 rectangular grid of e-bike parking spaces with no recharge facility (cost should be lower than that for a recharging space).

Customer specifications:

The implemented parking system should be able to provide facilities to:

- Maintain a database of registered drivers including their personal details including name, university ID number, address, phone number, account balance and staff or student status.
- Registered users should be able to pre-pay for their parking or to accumulate charges (up to a pre-set amount).

System Users

The system should be able to provide functionality for different users listed below:

- **Administrator** who will be able to:
  - Add/delete/edit drivers personal details
  - Credit a drivers account
  - Set parking charges
- **Entry system worker** who will be able to:
  - Record the entry of a registered driver by name.
  - Allocate a randomly chosen parking space to a car. For an e-bike, the driver should be asked if they would like a space with/without recharging and then a randomly chosen space is allocated.
  - View current car-park occupation and number of free spaces.
- **Exit system worker** who will be able to:
  - Free a previously allocated car slot
  - Charge the leaving customer accordingly.

o