ICT209 Assignment 2 2011

Objectives:

- Demonstrate that you can do Object Oriented design.
- Demonstrate that you can write Object Oriented programs using C++.
- Demonstrate that you can design and write programs using user defined data structures.
- Demonstrate that you can use data structures appropriately.
- Demonstrate that you can write test plans and show evidence of systematic testing.
- Demonstrate that you can draw UML diagrams.

You do not work in groups for this assignment as this is an individual assignment.

Worth:
30% of the unit

Due:
Midnight, end of Session 12 (last teaching week). See due date and time (and any amendments to these) in LMS.

How to submit (also see unit guide - section on Assignment/Project submission/return:

Non-Murdoch Campus:
Into the assignment submission area for the unit in LMS.

Murdoch Campus Internal students:
Into the assignment submission area for the unit in LMS **and** Assignment box located on Level 3 outside the School of IT office.

Externals:
Into the assignment submission area for the unit in LMS.

For submitting in LMS, zip up the entire folder. Make sure that you have included all needed files. Do not include temporary files or files not relevant to the assignment.

Name the zip file with the unit code, Assignment number, your name, student number.
ICT209Asg2JoBlogs12345678.zip
or alternatively,
ICT209_Asg2_JoBlogs_12345678.zip

Textual submissions should be type-written. External documentation can only be in the following formats:
Text (.txt)

PDF (.pdf)
RTF (.rtf)
HTML (.html)
Image formats : PNG, GIF, JPG, TIFF, BMP. (BMP and TIFF cannot be used for Web documents)

Assignment cover sheet requirements are listed in the unit outline found in the Admin area. Also see the end of this document.

Readings needed:

- Unit textbook. "*C++ Programming: Program Design Including Data Structures*" by D.S. Malik.
- Lecture notes.

Assignment Question:
Design an object-oriented solution and implement the solution in C++ to solve the problem described below. You must read the entire specification before starting work on this assignment.

Using programming exercises 5 and 6 from the end of Chapter 11 (Classes and Data Abstraction) in the unit textbook ("*C++ Programming: Program Design Including Data Structures*" 4th edition by D.S. Malik.), write a console based program to simulate a library where library patrons can borrow and return books.

Exercise 5 involves creating a book Abstract Data Type (ADT) called *bookType*. Exercise 6 involves creating *memberType*. However in this assignment, *memberType* would **not** have information on number of books bought and amount spent. Exercise 5 (book*Type*) is unchanged (will include price). Both *bookType* and *memberType* are used in this assignment.

You need to complete exercises 5 and 6, with exercise 6 modified as indicated above.

Exercises 5.c and 6.e involve testing the individual ADTs. Make sure you test any ADTs (that you create) thoroughly before incorporating them into the assignment. You must provide results of your testing and the test data used.

You need to work out the operations of a library from the point of view of a library patron. As a minimum, you need to design and implement these operations.

- Borrow books (patron)
- Return books (patron)
- Renew books (patron)
- Check current loans (patron)
- Add books to the library (library staff)
- Add patrons to the library (library staff)
- Exit (think of who can utilise this option and how it will be utilised; then design and implement)

Provide a simple text-based menu which enables the library patron to use your program. Library staff can add books and patrons to the library. Make sure you design a simple menu so that users do not get confused.

STL data structures can be used in this assignment. You must use the Binary search tree and the STL map (and/or multimap) data structures. You need to identify where these data structures can be used and provide a rationale for their use. There is a 30 marks penalty if you do not do this.

You may use std::string and string stream classes in your program instead of using C like strings. You may use iostream and file handling classes and objects in C++. See laboratory exercises.

Your program should be able to save and load data using files. You do not want to enter all the books and patron data every time your run your program. Data can be saved using the CSV format. For more information on this, see the laboratory exercise for session 5.

As borrowed books have a due date, you can use the date class from the laboratory exercises to keep track of the due dates. See laboratory exercises for session 4 onwards.

When working on this assignment, think of the usability of your program from the point of view of the user (patrons and library staff). This is the first thing you have to do, even before you start designing the program. This will form the menu system for the running program. You should not make the program more complicated than what is specified in the assignment. Make sure you provide an option to exit the program. It is not a good idea to exit a program by killing the program or by re-booting the computer.

Any advice and further clarifications to these requirements would be found in the readme file in the assignment 1 area. Questions and Answers (if any) would also be found in this readme file.

Documentation: *(printed versions apply only to internal students at the Murdoch campus.)*

- UML diagram showing the design of your classes. (printed and soft copy)
- Rationale for including **each** method and attribute in any class that you write. You need to also explain why you named **each** attribute and method in a particular way. It is not acceptable to say that is how others do it. (printed and softcopy)
- Rationale for the use of the tree and map data structures.
- A high level algorithm for the solution. (printed and softcopy)
- Doxygen output which shows all information as was done in the practice for week 2. (soft copy only)
- Test plan and output of the test run(s). (printed and soft copy)

Do not print code. Code will only exist as soft copy.

Minimum requirements:
You must provide all of the following; otherwise 50 marks will be deducted.

- UML diagram
- Written rationale for the design
- Doxygen output
- Program that builds (using Microsoft Visual C++ or code::blocks) and runs.
- Source code with doxygen style comments. (soft copy only)
- Test plan and output of test run(s)

- Executable program with associated data files in a separate directory called "*executable*". Make sure that the executable runs on a machine which does not have a compiler. Data files needed to enable it run should be included here.
- A declaration indicating what works and what does not work in your program. This can form the summary of your test plan and output of test runs and should be provided as a separate document called "***evaluation.txt***"

Marking

UML diagram	10%
Written rationale for the design + algorithm	20%
Program (includes coding style and comments)	55 %
Test plan and testing	15%

There is a 10 marks penalty if the cover sheet shown next is not filled in properly and/or the requirements of the coversheet are not met. If the cover sheet is not provided, you get no marks.

Penalty summary:

Specified data structures not used: -30 marks
Missing minimum requirements: -50 marks
Cover sheet problems: -10 marks

The lowest total mark you can get is 0.

MURDOCH UNIVERSITY PERTH, WESTERN AUSTRALIA *School of Information Technology*

ICT209 COVER SHEET

Given Names	Surname	Student Numbers	Mode (D/X)	Email

Name of tutor: _____________________________ **Day & Time of tutorial: _________________**

Assignment Number: _____ **Due Date: ______________** **Date Submitted: ______________**

If the given name by which your tutor knows you differs from your name on university records, you should indicate both names above. Tutor's name must be entered. Penalty is 10% of the total marks for the submission for not providing information asked for.

Your assignment should meet the following requirements. Please confirm this (by ticking boxes) before submitting your assignment that you have checked the declarations you need to make. ***Some items do not apply for online/electronic submission.***

☐ Except where I have indicated, the work I am submitting is my own work for the purpose of this assessment and has not been submitted for assessment in another unit.

☐ This submission complies with Murdoch University's academic integrity commitments. I am aware that information about plagiarism and associated penalties can be found at http://www.murdoch.edu.au/teach/plagiarism/. If I have any doubts or queries about this, I am further aware that I can contact my Unit Coordinator prior to submitting the assignment.

☐ I acknowledge and agree that the assessor of this assignment may, for the purpose of assessing this assignment:
- Reproduce this assignment and provide a copy to another academic staff member; and/or
- Submit a copy of this assignment to a plagiarism-checking service. This web-based service will retain a copy of this work for the sole purpose of subsequent plagiarism checking, but has a legal agreement with the University that it will not share or reproduce it in any form.

☐ I have retained a copy of all submitted work.

☐ I will retain a copy of the notification of receipt of this assignment. LMS submission would provide the required receipt. If you have not received a receipt within three days, please check with your Unit Coordinator.

☐ Assignment is presented on A4 size paper and is neatly collated. (Internal students)

☐ Assignment includes virus-free disk with machine-readable programs & files relevant only to this submission. CD or DVD only may be accepted. (Internal students)

☐ Instructions relating to answering of questions, formats used for electronic submission and LMS submission have been followed.

☐ Writing is clearly legible or has been printed.

☐ Pages have been firmly stapled. (Internal students)

____

Signature for submission via the assignment box (internal students)

For Submission in LMS:

☐ I am aware that I am making this declaration by submitting this document electronically and by using my Murdoch ID and password it is deemed equivalent to executing this declaration with my written signature.