# CSCI204 Assignment 1

## (Total 10 marks, Due by 11:59 pm sharp on Friday, 4th April, 2014)

---

**Aims**

This assignment aims to establish a basic familiarity with C++ classes. The assignment introduces increasingly object-based, C++ style of solution to a problem.

---

**General Requirements**

- Observe the common principles of OO programming when you design your classes.
- Put your name, student number at the beginning of each source file.
- Make proper documentation and implementation comments in your codes where they are necessary.
- Logical structures and statements are properly used for specific purposes.

---

**Objectives**

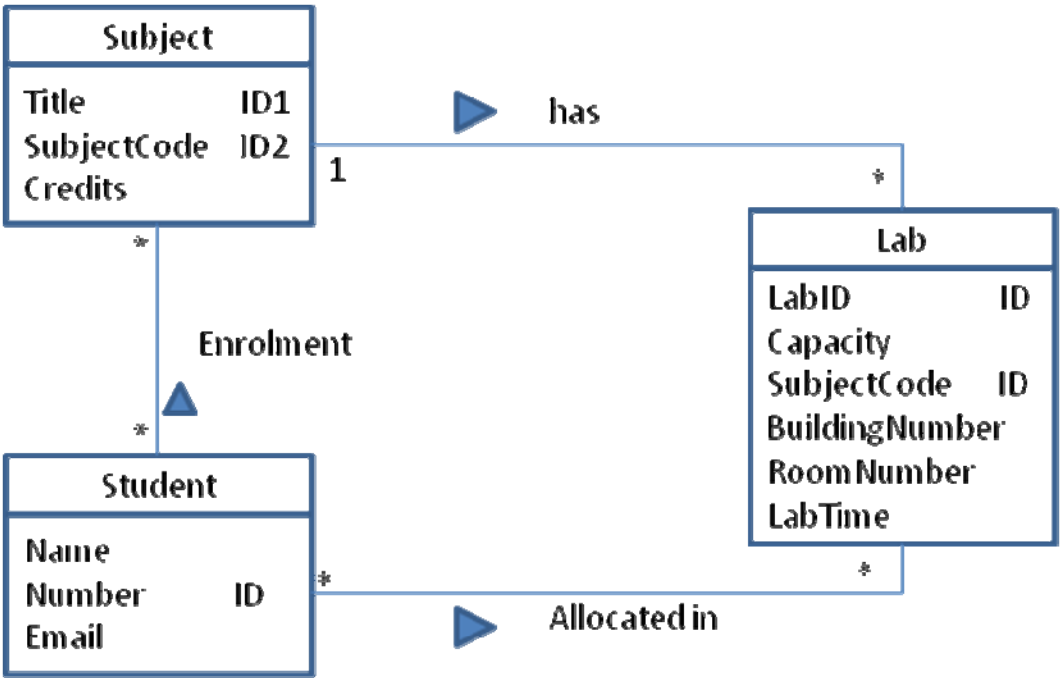
On completion of these tasks you should be able to:

- Code and run C++ programs using the development environment.
- Make effective use of the on-line documentation system that supports the development environment.
- Code programs using C++ in a hybrid style (procedural code using instances of simple classes) and in a more object-based style.
- Manipulate string data.

---

**Task:**

In this task, you will implement a C++ program to simulate students' lab enrollment. The UML diagrams are shown as following:




**Note: The attributes that marked ID, ID1, ID2 are the key attributes that can identify an object of those classes.**

You will define the classes **Subject**, **Student**, **Lab**, associated classes **Enrolment** and **Allocation** according to the UML diagrams above.

Define a class **Subject** in a file **Subject.h** with data members shown in the UML diagrams above. Implement necessary member functions (such as constructors, set and get functions, print function) in a file **Subject.cpp**.

Define a class **Student** in a file **Student.h** with data members shown in the UML diagrams above. Implement necessary member functions in a file **Student.cpp**.

Define a class **Lab** in a file **Lab.h** with data members shown in the UML diagrams above. Implement necessary member functions in a file **Lab.cpp**.

Define a class **Enrolment** in a file **Enrolment.h** with data member student number, subject code, enrolled date. Implement necessary member functions in a file **Enrolment.cpp**.

Define a class **Allocation** in a file **Allocation.h** with data member student number, lab ID and subject code. Define a class **AllocationNode** with an object of Allocation, and a pointer of AllocationNode in the file **Allocation.h**. Define a class **AllocationList** with two pointers of AllocationNode in the file Allocation.h. One pointer points to the head,

another pointer points to the tail of a linked list. The classes AllocationNode and AllocationList will be used to make a linked list to store the lab enrolments.

Implement necessary member functions in a file **Allocation.cpp** for the classes Allocation, AllocationNode and AllocationList. The member functions allow the program to create Allocation, AllocationNode objects (constructors or set functions), add new node at the end of the linked list (e.g. push_back(const Allocation &)), search lab allocation information by using pointers of the linked list.

Define a class Admin in a file **Admin.h** with data members:

- A pointer points to a dynamic array of Student objects;
- An integer stores the total number of Student objects;
- A pointer points to a dynamic array of Subject objects;
- An integer stores the total number of Subject objects;
- A pointer points to a dynamic array of Lab objects;
- An integer stores the total number of Lab objects;
- A pointer points to a dynamic array of Enrolment objects;
- An integer stores the total number of Enrolment objects;
- An object of a linked list for students' lab allocations.

Implement member functions described below in a file **Admin.cpp**:

- A function loads student's records from a given text file into a dynamic array of students.
- A function loads subject's records from a given text file into a dynamic array of subjects.
- A function loads lab's records from a given text file into a dynamic array of labs.
- A functions loads enrolment's records from a given text file into a dynamic array of enrolments.
- A function prints all students' information that loaded.
- A function prints all subjects' information that loaded.
- A function prints all labs' information that loaded.
- A function prints all enrolments' information that loaded.
- A function displays subjects' codes for a student.
- A function displays labs' IDs for a subject.
- A function manipulates lab allocations for students.
- A function prints out lab allocation list for each lab.
- Other necessary member functions.

Implement **main()** in a file **main.cpp** that gets command line inputs, declares an object of Admin, then call the member functions of **Admin** to load all records for students, subjects, labs and enrolments, display loaded records, manipulates students lab allocations by student's number, subject code and lab / tutorial ID. The manipulation function should verify the input values for the student's number, subject code and lab /

tutorial ID. At the end, the program will print lab enrolment list for each lab. The details can be found at Testing section.

**Note: Do not put statement "using namespace std;" in the head files. Do not define public data members for the classes.**

**Testing:**

In the working directory on banshee, you can compile the program by

CC –o ass1 main.cpp Admin.cpp Subject.cpp Student.cpp Enrolment.cpp Lab.cpp Allocation.cpp

Or to make it simple, you can compile the program by

CC –o ass1 *.cpp

To run the program, you should pass the file names to the main() by

./ass1 student_filename subject_filename lab_filename enrolment_filename

You can download the test files **subjects.txt**, **students.txt**, **labs.txt** and **enrolments.txt** from this site.

**Note: Do not define constant files' names inside the source code.**

You can use command **bcheck** on banshee to check if there is any memory leak by

bcheck ass1 student_filename subject_filename lab_filename enrolment_filename

In the following testing example, red data denote input data from keyboard.

```
 $ ./ass1 students.txt subjects.txt labs.txt enrolments.txt
Load students' records.....
1234567, Alice Montage, am12@uow.edu.au
1234568, Bob Smith, bs36@uow.edu.au
1234569, Cart Dong, cd28@uow.edu.au
1234570, Mark Twain, mt12@uow.edu.au
Load subjects' records.....
CSCI103, Algorithms and Problem Solving, 6
CSCI114, Procedural Programming, 6
CSCI124, Applied Programming, 6
CSCI203, Algorithms and Data Structures, 6
CSCI204, Object and Generic Programming in C++, 6
CSCI222, System Development, 6
CSCI235, Databases, 6
```

CSCI315, Database Design and Implementati, 6
CSCI321, Project, 12
Load labs' records.....
LabA, 40, CSCI204, 3, 127, Thursday 15:30-17:30
LabB, 40, CSCI204, 3, 127, Friday 8:30-10:30
LabA, 40, CSCI235, 3, 125, Monday 13:30-15:30
LabB, 40, CSCI235, 3, 125, Tuesday 17:30-19:30
TutorialA, 20, CSCI103, 3, 123, Tuesday 9:30-11:30
TutorialB, 20, CSCI103, 3, 122, Tuesday 9:30-11:30
TutorialC, 20, CSCI103, 1, G04, Tuesday 10:30-12:30
Load enrolments' records.....
CSCI103, 1234567, 25/02/2014
CSCI114, 1234567, 26/02/2014
CSCI124, 1234567, 27/02/2014
CSCI204, 1234567, 25/02/2014
CSCI222, 1234567, 27/02/2014
CSCI235, 1234567, 26/02/2014
CSCI315, 1234567, 25/02/2014
CSCI103, 1234568, 24/02/2014
CSCI114, 1234568, 27/02/2014
CSCI124, 1234568, 26/02/2014
CSCI204, 1234568, 23/02/2014
CSCI222, 1234568, 26/02/2014
CSCI235, 1234568, 28/02/2014
CSCI315, 1234568, 03/03/2014
CSCI321, 1234568, 28/02/2014

Student number (0-Exit): 12345
Student number 12345 not exists.
Student number (0-Exit): 1234567
Enrolled in the subjects: CSCI103 CSCI114 CSCI124 CSCI204 CSCI222 CSCI235 CSCI315
Subject code (0-Stop): CSCI321
Incorrect subject code.
Subject code (0-Stop): CSCI103
Available labs/tutorials: TutorialA TutorialB TutorialC
Lab ID: LabA
Student cannot be allocated in this lab/tutorial.
Student number (0-Exit): 1234567
Enrolled in the subjects: CSCI103 CSCI114 CSCI124 CSCI204 CSCI222 CSCI235 CSCI315
Subject code (0-Stop): CSCI103
Available labs/tutorials: TutorialA TutorialB TutorialC
Lab ID: TutorialA
Student number (0-Exit): 1234567

Enrolled in the subjects: CSCI103 CSCI114 CSCI124 CSCI204 CSCI222 CSCI235 CSCI315
Subject code (0-Stop): CSCI103
The student has already been allocated in a lab/tutorial for the subject.
Subject code (0-Stop): CSCI204
Available labs/tutorials: LabA LabB
Lab ID: LabB
Student number (0-Exit): 1234567
Enrolled in the subjects: CSCI103 CSCI114 CSCI124 CSCI204 CSCI222 CSCI235 CSCI315
Subject code (0-Stop): CSCI124
Available labs/tutorials: No lab / tutorial available.
Student number (0-Exit): 1234567
Enrolled in the subjects: CSCI103 CSCI114 CSCI124 CSCI204 CSCI222 CSCI235 CSCI315
Subject code (0-Stop): CSCI235
Available labs/tutorials: LabA LabB
Lab ID: LabB
Student number (0-Exit): 1234568
Enrolled in the subjects: CSCI103 CSCI114 CSCI124 CSCI204 CSCI222 CSCI235 CSCI315 CSCI321
Subject code (0-Stop): CSCI103
Available labs/tutorials: TutorialA TutorialB TutorialC
Lab ID: TutorialA
Student number (0-Exit): 1234568
Enrolled in the subjects: CSCI103 CSCI114 CSCI124 CSCI204 CSCI222 CSCI235 CSCI315 CSCI321
Subject code (0-Stop): CSCI103
The student has been allocated in a lab/tutorial for the subject.
Subject code (0-Stop): CSCI204
Available labs/tutorials: LabA LabB
Lab ID: LabA
Student number (0-Exit): 1234568
Enrolled in the subjects: CSCI103 CSCI114 CSCI124 CSCI204 CSCI222 CSCI235 CSCI315 CSCI321
Subject code (0-Stop): CSCI235
Available labs/tutorials: LabA LabB
Lab ID: LabB
Student number (0-Exit): 0

CSCI204--Object and Generic Programming in C++
Lab / Tutorial: LabA
1234568, Bob Smith, bs36@uow.edu.au

CSCI204--Object and Generic Programming in C++
Lab / Tutorial: LabB

1234567, Alice Montage, am12@uow.edu.au

CSCI235--Databases
Lab / Tutorial: LabA

CSCI235--Databases
Lab / Tutorial: LabB
1234567, Alice Montage, am12@uow.edu.au
1234568, Bob Smith, bs36@uow.edu.au

CSCI103--Algorithms and Problem Solving
Lab / Tutorial: TutorialA

CSCI103--Algorithms and Problem Solving
Lab / Tutorial: TutorialB

CSCI103--Algorithms and Problem Solving
Lab / Tutorial: TutorialC

**Note: Your program should work on different testing data.**

**Submission**

**This assignment is due by 11.59 pm (sharp) on Friday, 4th April, 2014.**

Assignments are submitted electronically via the ***submit*** system. For this assignment you must submit all the files via the command (in one line):

submit -u your_user_name -c CSCI204 -a 1 main.cpp Admin.h Admin.cpp Subject.h Subject.cpp Student.h Student.cpp Enrolment.h Enrolment.cpp Lab.h Lab.cpp Allocation.h Allocation.cpp

and input your password.

**Make sure that you use the correct file names. The UNIX system is case sensitive. You must submit all files in one *submit* command line.**

**Your program code must be in a good programming style, such as good names for variables, methods, classes, no global variable(s), and keep indentation.**

**Submission via e-mail is NOT acceptable.**

**After submit your assignment successfully, please check your email of confirmation. You should keep the email for the reference.**

**You would get 50% off the marks if your program codes could not be compiled correctly.**

Late submissions do not have to be requested. Late submissions will be allowed for a few days after close of scheduled submission (up to 3 days, include weekend). Late submissions attract a mark penalty (25% off the marks for each day late); this penalty may be waived if an appropriate request for Academic Consideration (for medical or similar problem) is made via the university SOLS system ***before*** the Due time. No work can be submitted after the late submission time.

A policy regarding late submissions is included in the course outline.

The assignment is an **individual assignment** and it is expected that all its tasks will be solved **individually without any cooperation** with the other students. If you have any doubts, questions, etc. please consult your lecturer or tutor during tutorial classes or office hours. Plagiarism will result in a **FAIL** grade being recorded for that assessment task.

# End of specification