KXC251 Algorithms and Metrics, ZUT 2011

Assignment 1

Due: 3pm on Tuesday 15th November, 2011

This assignment is worth 15% of your total mark for the unit.

Introduction

You are required to write a program for storing information about university students and units. Each student has a 6 digit student number (e.g. 061234), a name (e.g. Chen Jun), and is enrolled in multiple units (to be stored in a Binary Search Tree). For each unit, you need to store only a six character course code (e.g. kxc251).

Your program should display a menu and perform the following operations:

1. Add a student
2. Change student details
3. Enrol a student in a unit (adding a unit to a student)
4. Print the name and course codes of all units a particular student is enrolled in (units must be in alphabetical order)
5. Print the student numbers and names of all students in the course

Implementation

You should create a file called "assignment1.cpp" which contains the main function to run the program. The main function should display the menu when the program begins and when each of the operations finish.

You should create files for 2 classes:

- The student class should store the details of each student, plus a tree of units the student is enrolled in.
- The unit class should store the unit details.

You should also include a list class:

- You MUST use the list code provided in lectures, but you will need to modify it to store students.
  IMPORTANT: You are NOT allowed to use any other list class (for example, the STL list class)

You should also include a tree class:

- You MUST use the tree code provided in lectures, but you will need to modify it to store course codes.
  IMPORTANT: You are NOT allowed to use any other tree class (for example, the STL tree class)

Menu

You should use console output to create a simple menu interface for your program as shown below:

```
Welcome to UnitData
1) Add student
2) Change student details
3) Enrol a student in a unit
4) Display student details
5) Display all students
0) Exit

Please choose an option:
```

1. Add Student

If the user chooses ‘1’ the program should ask the user for the student number of the student and the name of the student. This is demonstrated in the example below:

```
Please enter the student’s number (eg. 940561):
061234

Please enter the student’s name:
Chen Jun
```

The program should store the student in its student linked list (you can assume that the user never tries to add the same student twice). The student linked list should be kept in sorted order at all time (increasing numerical order by student number).

2. Change Student Details

If the user chooses ‘2’ the program should ask the user for the student number of the student whose details are to be changed and the new student name. This is demonstrated in the example below:

```
Please enter the student’s number:
882425

Please enter the student’s new name:
Ian Lewis
```

The program should store the information then redisplay the menu.

3. Enrol Student in a Unit

If the user chooses ‘3’ the program should ask the user for a student number and a unit code. This is demonstrated in the example below:

```
Please enter the student’s number:
882425

Please enter the unit code to enrol them in:
kxc251
```

The program should store the information then redisplay the menu.

4. Display Student Details

If the user chooses ‘4’ the program should ask the user for a student number, and then output the student’s name and list all units the student is enrolled in (in increasing alphabetical order). This is demonstrated in the example below:

```
Please enter the student’s number:
123456

Wang Bo
kxa355, kxc251, kxt101, kxt103, kxt104
```

The program should then redisplay the menu.

5. Display All Students

If the user chooses ‘5’ the program should display the student numbers and student names of all students enrolled in the course. As shown below, each student must:

1) be displayed in order of their student number (from lowest to highest), and
2) only occur once in the list (i.e. no duplicates).

```
123456, Wang Bo
234567, Zhao Jie
882425, Ian Lewis
```

The program should then redisplay the menu.

6. Display Enrolment Details

If the user chooses ‘6’ the program should ask the user for a unit code, and then display the student numbers of all students enrolled in that unit. This is demonstrated in the example below:

```
Please enter the unit code:
kxc251

123456, 882425
```

The program should then redisplay the menu.

0. Exit

If the user chooses ‘0’ the program needs to correctly free ALL dynamically allocated memory and close the program.

Points to Note

All strings should be stored efficiently. This means that you will need to read stings into a buffer and then dynamically allocate the appropriate amount of memory for the string, as shown in lectures.

Your program does not need to display error messages for incorrect user input — you may assume all input is entered correctly. However, you may find that it is easier to test your program if you have some error checking code.

Submission

1. You should submit an electronic copy of your program to your Chinese lecturer, including the following files:

   assignment1.cpp (containing the “main” function)

   studentList.h

   studentList.cpp

   unitTree.h

   unitTree.cpp

   student.h

   student.cpp

   assignment1.cpp (containing the “main” method)

   Your files should be well documented. If you have used code from the lectures, or from anywhere else, this should be clearly stated at the top of the file.

2. You should submit a signed Assignment Cover Sheet to your Chinese Lecturer.

Marking

Your program will be executed and tested for correctness.

Your program may be tested by compiling and running it from a command line interface. Please ensure that the main file “assignment1.cpp” is named correctly.

Your code will also be examined to determine a mark for coding style (e.g. comments, suitable indentation, structure etc). The following marking sheet (which will be returned to you) shows the mark allocation for each part of the assignment.

Assignment 1

Student Name: **Student #:**

Item	**Marks**
Program functions correctly:	
• Program starts and terminates correctly	/1
• Add Student	/1
• Edit Student Details	/2
• Enrol Student in Unit	/1
• Display all students	/1
• Display a single student's details and enrolments	/3
• Display all student enrolments in a single unit	/3
Program style:	
• Program contains appropriate classes and methods	/1
• Dynamically allocated memory is dealt with correctly	/1
• Code is well documented and formatted	/1
Total	**/15**

Comments:____

____

____

____

____

____