# UNIVERSITY OF WOLLONGONG

## SCHOOL OF COMPUTER SCIENCE AND SOFTWARE ENGINEERNG

CSCI124 APPLIED PROGRAMMING SPRING 2013

ASSIGNMENT 1 (5 MARKS)

DUE: 11:59PM FRIDAY 23 AUGUST 2013 (WEEK 4)

## AIM

Basic revision on developing programs via functional decomposition

On completion of the assignment, you should know how to:

- Design a program by decomposing the problem into smaller functional units
- Implement a program in stages
- Produce a C++ database program for maintaining records

## REQUIREMENTS

In this task, you are going to develop a database program to store information related to the students of a university. Information to be stored related to a student

Each student record in the database contains the following information.

| | |
|---|---|
| Name | at most 30 char (cannot contain the newline character, can contain anything else but not NULL) |
| ID | a unique identifier a positive integer of 7 digits (cannot be NULL) |
| Course | 4 digit course code, a space, then at most 80 char<br>e.g. 1612 Master of Computer Studies (cannot be NULL) |
| List of Subjects | each subject consists of the following information:<br>1. year (4 digits)<br>2. session (one char A/S for autumn/ spring)<br>3. code (7 characters)<br>4. credit (2 digits, i.e., if the credit point is 6, it will be represented a 06)<br>5. Mark (0 to 100 or e or w)<br>e enrolled, w means withdraw<br>You may assume each student can be associated with at most 30 subjects |
| GPA | the system should be able to calculate the GPA automatically based on credit and mark of the subject list of the student |

GPA is defined as:

Sum of {credit of all subjects multiplied by their scores} divided by Sum of {credits of all subjects}

All subjects here means subjects that is not of the status e and w.

A sample data file (a plain text file) is given below to illustrate the format.

The ###START### and ###END### are used to mark the starting and ending of the file, while ###RECORD### is used to illustrate the starting of a new record.

```
###START###
###RECORD###
John Ng
1234567
1612 Master of Computer Studies
2012SCSCI12406w
2013ACSCI1240644
2013SCSCI1240699
###RECORD###
Alice Peterson
2412347
790 Bachelor of Engineering - Bachelor of Computer Science
2013ACSCI2050670
###END###
```

Your database program supports the following commands (the should be shown in the first level menu)

**Main Menu**

| | |
|---|---|
| | Showing the current number of records |
| q | Quit the program |
| i | import data from text file<br>// the function asks for a file name and import all records from the file<br>// IF record with duplicated ID exists, your program should display the information of both records and ask the user to choose one |
| e | export data to text file<br>// the function asks for a file name and output all records to this file<br>// IF the file already exists, your program should ask if the user would like to overwrite the file or cancel |
| s | Display the search menu. Details of the search menu are described below. |

You can assume the maximum size of your database is 1000. The import function will take in the maximum allowable records, return to the main menu and display the number of imported records.

**Search Menu**

The following commands are supported in the search menu

| | |
|---|---|
| | Showing the current number of records |
| r | return to main menu |
| n | Name search (case insensitive)<br>the function asks for a name (support one single wild card *)<br>* means everything, e.g. *n means any name that ends with the letter n and that A*n means any name that starts with A and ends with n |
| s | Subject Search (case sensitive) |

| | |
|---|---|
| | return a list of students taking the subject matching the search criteria<br>promote the user for year, session and code<br>for each input, the user can use the wild card * |
| g | Search by GPA, the input format is a symbol <, > or = followed by a floating point number.<br>for =, any result within 1 point of the specified GPA is considered a match<br>That is, if the user want to search for students with GPA = 5.5, it will display all students whose GPA is from 4.5 to 6.5 (inclusive) |
| NOTES | For command n, s and g, if the search do not return any record, your program should display "no record found" and stay in the search menu. Otherwise, your program should show the display menu described below. |

**Display Menu**

The following commands are supported in the search menu

| | |
|---|---|
| | Display the current record and the total number of records satisfying the search criteria |
| r | return to main menu |
| s | return to search menu |
| n | Go to the next record (this option is available only when the next record exists) |
| p | Go to the previous record (this option is available only when the previous record exists) |
| e | Edit the current record.<br>Upon completion, it shows the old and edited record and ask the user for confirmation.<br>After that, it returns to the search menu.<br>For all options, if the user just press enter, it means keeping the existing value.<br>If the newly entered ID is not unique, it gives an error and ask the user to re-enter the value. |
| d | Delete the current record.<br>Ask the user to confirm the deletion. If confirmed, it deletes the record and return to the main menu. |

All commands are case sensitive. If the user enters an invalid input, your system should prompt the user to re-enter. You can assume the user will never enter anything longer than 100 characters.

## GUIDELINES

Do not alter the menu options or input data requirements as they will be used to test your program. Comment your code appropriately.

Implement your program in stages. In the first stage, think of an appropriate menu structure. Next, implement the read file function. In order to test the read function, you may write another "display" function which simply displays all records in memory.

Do not hesitate to ask your lecturer or lab tutor if you are unsure of anything or encounter any difficulties. (Remember it is typically the software developer's responsibility to clarify the requirements of the software with the client.)

**Important: YOU MUST SUBMIT A MAKE FILE that compiles your code on Banshee. Your code should compile without any warnings.**

**To show all the warnings, you should compile with the -Wall option.**

## SUBMISSION

Submit your program (and Makefile, if any) via the submit command.

submit –u userid –c CSCI124 –a ass1 filenames

Please make sure you receive the submission receipt after submitting your files.

Remember that you have to put the following information on the header of each source file you will be submitting in this assignment:

- Student name
- Student number
- Lab

An extension of time for the completion of the assignment may be granted in certain circumstances. A request for an extension must be made to the Subject Coordinator before the due date. Supporting documentation must accompany the request for extension. Late assignments without granted extension will be marked but the mark awarded will be reduced by 1 mark for each day late. Assignments more than 3 days late will not be accepted.

## ASSIGNMENT 2 PREVIEW

To remind you the importance of program design and modular programming, assignment 2 will be based on assignment 1. The following are a preview of the additional functionalities required in assignment 2.

**Additional Requirements (all in the main menu) in Assignment 2**

| | |
|---|---|
| I | Import from a binary file. The format of a binary file is under your decision. |
| E | Export to a binary file. The format of the binary file is under your decision. The only requirement is that it has to be compatible with the Import from a binary file function. |
| V | Validate records. Check for any mismatch between subject code and subject for subjects in the same year at the same session. Display the conflicting records and ask the user to choose one of the two options:<br>- ignore<br>- all records to one of these conflicting names |
| O | Options. Choose the order of display. Sort key includes:<br>Name (alphabetical order), ID, Course Code, Course Name, GPA together with the priority.<br>(E.g., sort by name, if same name, then sort by ID OR sort by Course Code, if same course code, sort by GPA.)<br>I.e., priority for each criteria, and the method which include ascending and descending/<br>Records should always be displayed according to the option.<br>Default option is sort by ID (since ID is unique, the other does not matter…) |