

Sabancı University

Faculty of Engineering and Natural Sciences

CS204 Advanced Programming

Fall 2015-2016

Homework 2 – Students and Courses

Due: 12/10/2015, 10:00pm

PLEASE NOTE:

Your program should be a robust one such that you have to consider all relevant programmer mistakes and extreme cases; you are expected to take actions accordingly!

You HAVE TO write down the code on your own.

You CANNOT HELP any friend while coding.

Plagiarism will not be tolerated!

Introduction

The aim of this homework is to get familiar with basic manipulation with pointers and simple/double linked lists, such as: finding (searching for) a node in a list, dynamically adding/deleting a node in a list, creating new list, printing a list, printing a certain node in a list, as well as cooperation of both of those kinds of lists when building a more complex frameworks of linked lists of (double) linked list. Your job here is to manage the student's data that SU has for the term. Concretely, which student has been registered for which course(s). Since neither the number of students, nor the course(s) he/she will take are known beforehand, we will have to manage this information using dynamically, thus you will dynamically allocate and free memory on the run in accordance to your needs. This means that you are going to use the new statement when allocating a block of memory from the heap and the delete one when freeing a memory block you want need anymore. Pay attention to avoid memory leaks (unused memory which has not been freed) as well as accessing already freed (deleted) memory chunks.

Input file

SU decided it's enough for them to keep records of their students for the FALL 2015/16 term in a notepad file. You are asked to develop an application that will help SU student's recourse desk to better organize their job in maintaining and manipulating (printing, adding, deleting, searching, ...) with students data. The input file is shown below.

A record for a student starts with # proceeded by his/hers unique ID (e.g. #189754). After a comma, student's full name is given (e.g. John Smith). Then after a colon (:), student's course(s) he has been registered to for the FALL 2015/16 terms are given, each in a single row, until a full stop is reached. Courses are separated by comma with each other. Each registered course starts with its CRN code consisted of the department abbreviation, an underline and a three digit number (e.g. CS_204, IE_303). After an empty space, course's name is given. In the end, course's instructor is given in parenthesis (e.g. Albert Levi).

```

#15623, Canberk Ozturk: CS_201 Intro to Computing (Albert Levi),
                        CS_303 Logical&Digital System Design (Erkay Savas),
                        CS_306 Database Systems (Kamer Kaya),
                        CS_408 Computer Networks (Albert Levi),
                        EE_303 Analog Integrated Circuits (Yasar Gurbuz).

#189754, John Smith: CS_204 Advanced Programming (Kamer Kaya),
                    CS_303 Logical&Digital System Design (Erkay Savas),
                    CS_408 Computer Networks (Albert Levi),
                    CS_307 Operating Systems (Yucel Saygin),
                    CS_404 Artificial Intelligence (Berrin Yanikoglu),
                    MATH_201 Linear Algebra (Canan Kasikci).

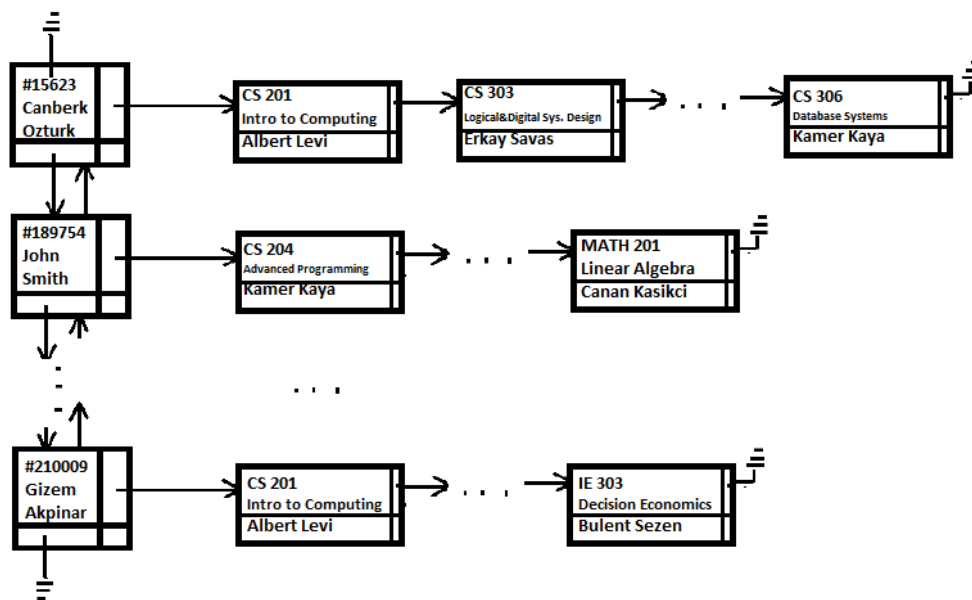
.....

#210009, Gizem Akpinar: CS_201 Intro to Computing (Albert Levi),
                       CS_303 Logical&Digital System Design (Erkay Savas),
                       ENS_203 Electronic Circuits (Meric Ozcan),
                       IE_303 Decision Economics (Bulent Sezen).

```

Data structure

In the beginning you are asked to extract the data from the input file and construct a data structure consisted of single linked lists of registered courses on a double linked lists of students, as it is illustrated in the figure below.



For each course node you are asked to store its code, courses name and instructor's full name. For each student you should store student's unique ID and his/hers full name.

While the simple linked lists of courses can be constructed on any order (not necessarily sorted), the double linked list of students **should always be sorted** (say in ascending order) with respect to the students ID (since it might happen multiple students to have same name and surname, but not a same ID). Knowing that there might be thousands of students, maintaining a sorted double linked list for students at any stage would improve performances when searching for and/or editing certain student's data. We expect from you to utilize this fact when writing your assignment. You should also have in

mind at any stage that if a student hasn't registered for any course during the term, he/she should be deleted from the doubly linked list.

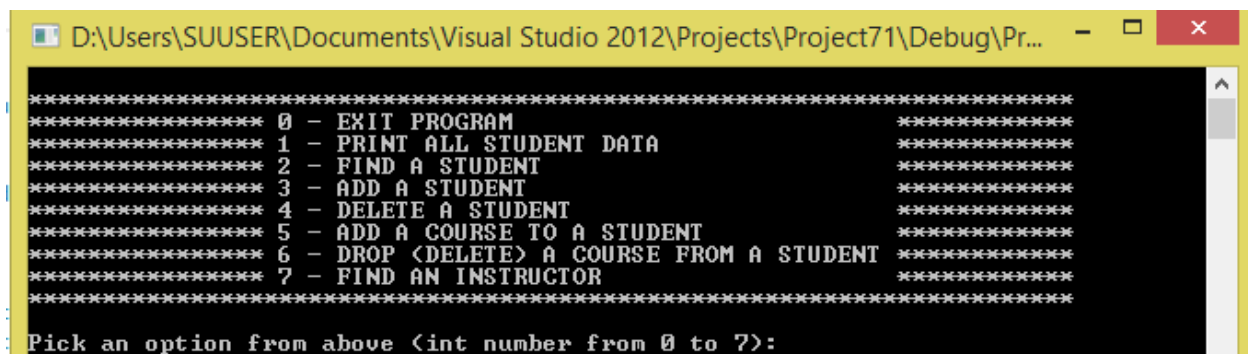
The main menu

Below we give to you the node structure for the double and the single linked list. We also show to you a print screen of the main menu, as well as a part of code of the main menu. It should give you a clue how to proceed with each menu option and the corresponding function that implements the option. Of course, you preserve the freedom to add/delete/change functions as you wish, given that the output is the desired one.

```
struct termCourse
{
    string courseName;
    string CRN;
    string instructor;
    termCourse * next;
};

struct student
{
    string studentName;
    int id;
    student * up;
    student * down;
    termCourse * right;
};

student *head = nullptr;
```



```
D:\Users\SUUSER\Documents\Visual Studio 2012\Projects\Project71\Debug\Pr...
*****
***** 0 - EXIT PROGRAM *****
***** 1 - PRINT ALL STUDENT DATA *****
***** 2 - FIND A STUDENT *****
***** 3 - ADD A STUDENT *****
***** 4 - DELETE A STUDENT *****
***** 5 - ADD A COURSE TO A STUDENT *****
***** 6 - DROP (DELETE) A COURSE FROM A STUDENT *****
***** 7 - FIND AN INSTRUCTOR *****
*****
Pick an option from above (int number from 0 to 7):
```

```
while (true){
    cout << endl;
    cout<< "*****" << endl;
    << "***** 0 - EXIT PROGRAM *****" << endl;
    << "***** 1 - PRINT ALL STUDENT DATA *****" << endl;
    << "***** 2 - FIND A STUDENT *****" << endl;
    << "***** 3 - ADD A STUDENT *****" << endl;
    << "***** 4 - DELETE A STUDENT *****" << endl;
    << "***** 5 - ADD A COURSE TO A STUDENT *****" << endl;
    << "***** 6 - DROP (DELETE) A COURSE FROM A STUDENT*****" << endl;
    << "***** 7 - FIND AN INSTRUCTOR *****" << endl;
    << "*****" << endl;
    cout << endl;
    int option;
    cout << "Pick an option from above (int number from 0 to 7): ";
    cin>>option;
    switch (option)
```

```

{
    case 0:
        cout<<"PROGRAM EXITING ... "<<endl;
        system("pause");
        exit(0);
    case 1:
        printStudentList();
        break;
    case 2:
        printStudent(findAStudent());
        break;
    case 3:
        addAStudent();
        break;
    case 4:
        deleteAStudent();
        break;
    case 5:
        addACourse();
        break;
    case 6:
        deleteACourse();
        break;
    case 7:
        findInstructor();
        break;
    default:
        cout<<"INVALID OPTION!!! Try again"<<endl;
} //switch
} //while (true)

```

Below we describe each of the main menu options (with option 0 being obvious and already implemented).

1) Print All Student Data

It prints all the recorded students and their corresponding term courses they are registered to. The output should be as shown below

```

Pick an option from above (int number from 0 to 7): 1
ID: 156234, Name&Surname: Canberk Ozturk, Taken courses in FALL 2015/16 :
    CS_201 Intro to Computin (Albert Levi)
    CS_303 Logical&Digital System Desig (Erkay Savas)
    CS_306 Database System (Kamer Kaya)
    CS_408 Computer Network (Albert Levi)
    EE_303 Analog Integrated Circuit (Yasar Gurbuz)

ID: 189754, Name&Surname: John Smith, Taken courses in FALL 2015/16 :
    CS_204 Advanced Programmin (Kamer Kaya)
    CS_303 Logical&Digital System Desig (Erkay Savas)
    CS_408 Computer Network (Albert Levi)
    CS_307 Operating System (Yucel Saygin)
    CS_404 Artificial Intelligenc (Berrin Yanikoglu)
    MATH_201 Linear Algebr (Canan Kasikci)

ID: 210009, Name&Surname: Gizem Akpinar, Taken courses in FALL 2015/16 :
    CS_201 Intro to Computin (Albert Levi)
    CS_303 Logical&Digital System Desig (Erkay Savas)
    ENS_203 Electronic Circuit (Meric Ozcan)
    IE_303 Decision Economic (Bulent Sezen)

```

2) Find A Student

This feature should allow the user to search for and find a student (if he/she exists). The search should be done using either the student's ID or his name.

If a student is not found, a proper text is printed and the program should send us back to the main menu, as it is shown bellow.

```
Pick an option from above <int number from 0 to 7>: 2
Enter students ID or Name&Surname:565423
STUDNET WAS NOT FOUND!!! GOING BACK TO MAIN MENU.

*****
***** 0 - EXIT PROGRAM *****
***** 1 - PRINT ALL STUDENT DATA *****
***** 2 - FIND A STUDENT *****
***** 3 - ADD A STUDENT *****
***** 4 - DELETE A STUDENT *****
***** 5 - ADD A COURSE TO A STUDENT *****
***** 6 - DROP <DELETE> A COURSE FROM A STUDENT *****
***** 7 - FIND AN INSTRUCTOR *****
*****

Pick an option from above <int number from 0 to 7>: 2
Enter students ID or Name&Surname:Artrim Kjamilji
STUDNET WAS NOT FOUND!!! GOING BACK TO MAIN MENU.
```

If a search is done giving the name&surname of the student we want to find, you should take care to give it as a single input (e.g. Artrim Kjamilji in this case should be taken as a single string). (Hint: use `getline(cin, name)`). If the student is found, his/hers information is printed in the following manner

```
Pick an option from above <int number from 0 to 7>: 2
Enter students ID or Name&Surname:199681
Student was found. ID:199681. Name&Surname: Astrit Hyka
CS_204 Advanced Programmin <Kamer Kaya>
CS_306 Database System <Kamer Kaya>
CS_307 Operating System <Yucel Saygin>
CS_412 Machine Learnin <Berrin Yanikoglu>
CS_408 Computer Network <Albert Levi>
MATH_201 Linear Algebr <Canan Kasikci>

*****
***** 0 - EXIT PROGRAM *****
***** 1 - PRINT ALL STUDENT DATA *****
***** 2 - FIND A STUDENT *****
***** 3 - ADD A STUDENT *****
***** 4 - DELETE A STUDENT *****
***** 5 - ADD A COURSE TO A STUDENT *****
***** 6 - DROP <DELETE> A COURSE FROM A STUDENT *****
***** 7 - FIND AN INSTRUCTOR *****
*****
```

3) Add A Student

This option adds a new student in the list. It first asks for the new student's ID (an integer). Knowing that ID's are unique, if an already existing ID is entered, a proper message is displayed and the user is returned in the main menu.

```
Pick an option from above <int number from 0 to 7>: 3
Give new student's ID: 210009
THE STUDENT ALREADY EXISTS. GIONG BACK TO MAIN MENU...

*****
***** 0 - EXIT PROGRAM *****
***** 1 - PRINT ALL STUDENT DATA *****
***** 2 - FIND A STUDENT *****
***** 3 - ADD A STUDENT *****
***** 4 - DELETE A STUDENT *****
***** 5 - ADD A COURSE TO A STUDENT *****
***** 6 - DROP <DELETE> A COURSE FROM A STUDENT *****
***** 7 - FIND AN INSTRUCTOR *****
*****
```

If a unique ID is entered, the program asks for the student's full name and surname (again, given as a single line parameter with multiple parameters). Afterwards, the user is asked whether he wants to add a course for the student. If yes (Y/y) is chosen, the user is prompted to enter course's CRN code, name

(as a single line parameter) and instructor. In the same manner the user can enter multiple courses, as shown below. We should keep in mind that the student's list is always sorted by their ID.

```
Pick an option from above <int number from 0 to 7>: 3
Give new student's ID: 199632
Enter student's name and surname: Ozgur Pular
To enter a new course for the student press <Y/y> for Yes.
Otherwise press any key
y
Enter course's CRN:CS_201
Give the name of the course:Intro to Computing
Give the name and surname of the instructor:Albert Levi
To enter a new course for the student press <Y/y> for Yes.
Otherwise press any key
y
Enter course's CRN:CS_306
Give the name of the course:Database Systems
Give the name and surname of the instructor:Kamer Kaya
To enter a new course for the student press <Y/y> for Yes.
Otherwise press any key
n
```

If for a newly entered student the user doesn't add any course, the student is going to be deleted (if a student hasn't any registered course, he should be removed from the list)

```
Pick an option from above <int number from 0 to 7>: 3
Give new student's ID: 568794
Enter student's name and surname: Alper Cetinkaya
To enter a new course for the student press <Y/y> for Yes.
Otherwise press any key
n
Student with ID: 568794 hasn't registered to any course for this term.
<S>HE WILL BE DELETED
```

4) Delete A Student

Deleting is exclusively done by ID (since there might more students with the same name and surname). If an entered ID doesn't exist a proper message is shown and program prints the main menu. If the ID is found, firstly student's courses are deleted and then his node. In the end a proper message is shown (given below). The newly freed memory should be added to the heap.

```
Pick an option from above <int number from 0 to 7>: 4
Give student's ID to be deleted: 210009
STUDENT WITH ID 210009 WAS DELETED. GOING BACK TO MAIN MENU...
```

5) Add a Course

First a prompt is shown asking for the students ID. If it doesn't exist, the user is returned to the main menu. If it exists the user is asked whether he wants to add a course to the student by choosing (Y/y) for yes or any character for no. If yes is chosen another prompt asks from the user to give the CRN code of the course he wants to add. If it already exists, a message is displayed telling this, as it is shown below, and the course will not be added. If the CRN code doesn't exist, the user is asked to also give course's and instructor's name (as a single line string) and afterwards the course is added to the student. The (Y/y) is repeated after each successful or unsuccessful attempt to add a course. This is illustrated in the figure below.

```
Pick an option from above <int number from 0 to 7>: 5
Give students ID you want to add a course for:
199545
Student was found. ID:199545, Name&Surname: Haddad Al-Akhli
To enter a new course for the student press <Y/y> for Yes.
Otherwise press any key
y
Enter course's CRN:CS_204
The course already exists
To enter a new course for the student press <Y/y> for Yes.
Otherwise press any key
y
Enter course's CRN:CS_201
Give the name of the course:Intro to computing
Give the name and surname of the instructor:Albert Levi
To enter a new course for the student press <Y/y> for Yes.
Otherwise press any key
n
```

6) Drop (Delete) A Course

Firstly, the user is asked to enter the ID of the student we wish to delete a course for. If it doesn't exist, we are sent back to the main menu. If it exists, the CRN code of the code we wish to delete is asked. If it doesn't exist, we are returned to the main menu. Those scenarios are shown below.

```
Pick an option from above <int number from 0 to 7>: 6
Give students ID you want to delete a course for:
58455
STUDNET WAS NOT FOUND!!! GOING BACK TO MAIN MENU.

*****
***** 0 - EXIT PROGRAM *****
***** 1 - PRINT ALL STUDENT DATA *****
***** 2 - FIND A STUDENT *****
***** 3 - ADD A STUDENT *****
***** 4 - DELETE A STUDENT *****
***** 5 - ADD A COURSE TO A STUDENT *****
***** 6 - DROP (DELETE) A COURSE FROM A STUDENT *****
***** 7 - FIND AN INSTRUCTOR *****
*****

Pick an option from above <int number from 0 to 7>: 6
Give students ID you want to delete a course for:
199545
Student was found. ID:199545, Name&Surname: Haddad Al-Akhli
Give the code for the course you want to drop for the student in this term:
IE_505
THE COURSE WAS NOT FOUND. GOING BACK TO MAIN MENU..
```

If an existing ID and CRN are entered, the course will be deleted, a proper message will be shown and the user will be send to the main menu, as it is shown below

```
Pick an option from above <int number from 0 to 7>: 6
Give students ID you want to delete a course for:
199545
Student was found. ID:199545, Name&Surname: Haddad Al-Akhli
Give the code for the course you want to drop for the student in this term:
CS_204
Course found. Deleting it...
GOING BACK TO MAIN MENU...

*****
***** 0 - EXIT PROGRAM *****
***** 1 - PRINT ALL STUDENT DATA *****
***** 2 - FIND A STUDENT *****
***** 3 - ADD A STUDENT *****
***** 4 - DELETE A STUDENT *****
***** 5 - ADD A COURSE TO A STUDENT *****
***** 6 - DROP (DELETE) A COURSE FROM A STUDENT *****
***** 7 - FIND AN INSTRUCTOR *****
*****
```

However, after the deletion if the student has no more courses, he/she should be also deleted from the list and a message should be shown, as it is shown below

```
Pick an option from above <int number from 0 to 7>: 6
Give students ID you want to delete a course for:
199632
Student was found. ID:199632, Name&Surname: Ozgur Pulur
Give the code for the course you want to drop for the student in this term:
CS_201
Course found. Deleting it...
GOING BACK TO MAIN MENU...
Student with ID: 199632 hasn't registered to any course for this term. <S>He WILL BE DELETED
```

7) Find Instructor

A prompt asking for instructors name and surname (as a single line string variable) should be shown in the beginning. If the entered lecturer is an instructor to a student (at least for one course) then the student should be printed. If the instructor gives more than one lecture to a student, he/she should be printed only once (no repetition is allowed). If the instructor cannot be found (either he doesn't exist or doesn't lecture for any of the term courses), a proper message should be displayed. Those cases are illustrated below

```

Pick an option from above <int number from 0 to 7>: 7
Give instructors name and surname:
Albert Einstein
The instructor Albert Einstein doesn't exist or doesn't lecture for this term.
GOING BACK TO MAIN MENU...
*****
***** 0 - EXIT PROGRAM *****
***** 1 - PRINT ALL STUDENT DATA *****
***** 2 - FIND A STUDENT *****
***** 3 - ADD A STUDENT *****
***** 4 - DELETE A STUDENT *****
***** 5 - ADD A COURSE TO A STUDENT *****
***** 6 - DROP <DELETE> A COURSE FROM A STUDENT *****
***** 7 - FIND AN INSTRUCTOR *****
*****
Pick an option from above <int number from 0 to 7>: 7
Give instructors name and surname:
Kamer Kaya
During this term, prof.Kamer Kaya lectures the following student(s):
    Canberk Ozturk
    John Smith
    Bridgete de Sanctis
    Mehmet Kaya
    Haddad Al-Akhli
    Astrit Hyka
GOING BACK TO MAIN MENU...

```

Some Important Rules

In order to get a full credit, your programs must be efficient and well presented, presence of any redundant computation or bad indentation, or missing, irrelevant comments are going to decrease your grades. You also have to use understandable identifier names, informative introduction and prompts. Modularity is also important; you have to use functions wherever needed and appropriate.

When we grade your homeworks we pay attention to these issues. Moreover, in order to observe the real performance of your codes, we are going to run your programs in *Release* mode and **we may test your programs with very large test cases**.

What and where to submit (PLEASE READ, IMPORTANT)

You should prepare (or at least test) your program using MS Visual Studio 2012 C++. We will use the standard C++ compiler and libraries of the abovementioned platform while testing your homework. It'd be a good idea to write your name and last name in the program (as a comment line of course).

Submissions guidelines are below. Some parts of the grading process are automatic. Students are expected to strictly follow these guidelines in order to have a smooth grading process. If you do not follow these guidelines, depending on the severity of the problem created during the grading process, 5 or more penalty points are to be deducted from the grade.

Name your cpp file that contains your program as follows:

“SUCourseUserName_YourLastname_YourName_HWnumber.cpp”

Your SUCourse user name is actually your SUNet username that is used for checking sabanciuniv e-mails. Do NOT use any spaces, non-ASCII and Turkish characters in the file name. For example, if your SUCourse user name is cago, name is Çağlayan, and last name is Özbugsizkodyazaroglu, then the file name must be:

Cago_Ozbugsizkodyazaroglu_Caglayan_hw2.cpp

Do not add any other character or phrase to the file name. Make sure that this file is the latest version of your homework program. Compress this cpp file using WINZIP or WINRAR programs. Please use "zip" compression. "rar" or another compression mechanism is NOT allowed. Our homework processing system works only with zip files. Therefore, make sure that the resulting compressed file has a zip extension. Check that your compressed file opens up correctly and it contains your cpp file.

You will receive no credits if your compressed zip file does not expand or it does not contain the correct file. The naming convention of the zip file is the same as the cpp file (except the extension of the file of course). The name of the zip file should be as follows:

SUCourseUserName_YourLastname_YourName_HWnumber.zip

For example zubzipler_Zipleroglu_Zubeyir_hw1.zip is a valid name, but

hw1_hoz_HasanOz.zip, HasanOzHoz.zip

are **NOT** valid names.

Submit via SUCourse ONLY! You will receive no credits if you submit by other means (e-mail, paper, etc.).

Successful submission is one of the requirements of the homework. If, for some reason, you cannot successfully submit your homework and we cannot grade it, your grade will be 0.

Good Luck! Please use discussion board at SUCourse for HW related questions.

CS204 Team (Artrim Kjamili)