ENSF 337- Programming Fundamentals
Term Project Instructions

Department of Electrical & Computer Engineering
University of Calgary
Maan Khedr

Program outputs and prompts:

This section provides the program output screen and expected functionality based on the text files provided for input

Main screen upon launching

Upon launching the program the program will show a welcome screen that provides the software information as such:

```
Student Registration System: V1.0
Development Team: Maan Khedr
Release date: June 7th, 2021

<<<<<< Press Enter to continue >>>>>>
```

To get the program to clear the terminal screen whenever you want, you will use the instruction:

System("clear");

The system statement lets you send a command to your OS, in this case we are sending the command clear to the terminal which clear the terminal screen.

Note that development team will be the members of the group/groups contributing to the program development and not the instructor name.

Upon pressing Enter, the program will carry on to attempt to load the files provided to the program as command args. You will need to create two empty vectors of types course and student, and you will populate the vectors with the information provided in the files during program execution call.

To load the files you will use the functions ***load_coursefile*** and ***load_studentfile*** with prototypes provided in the ***program_functions.h*** file. These two functions load information from the course and student text files respectively, where the file format is assumed to be as follows:

- Courses.txt:

Course_id start_year start_month start_day end_year end_month end_day class_capacity

- Students.txt:

Student_id First_name Last_name phone_number birth_year birth_month birth_day

The two functions will load the information without duplicating entries. For example in the courses.txt file the course id ENSF337 appears twice but the function should only create one course and skip the second time it runs into the course id. To do so you will use the find_course and find_student functions to check if course id is already in the vector of courses created so far and student id is in the vector of students created so far. Upon completion of loading the files the program will print the number of students and courses loaded from the files. The output will look as such:

~/project

```
4 Courses were loaded from courses file
3 Students were loaded from students file
<<<<<< Press Enter to continue >>>>>>
```

Note that we would want the program to clear screen after each prompt to continue.

If the program was not provided exactly 2 files to load as args in the command line, we will discard the provided file and show an output message stating wrong input files

```
~/project
Wrong number of inputs

<<<<<< Press Enter to continue >>>>>>
```

Main menu

After successful loading of files provided or failure due to incorrect input to program, the program will go into an infinite loop (***while(true)***) and will show the user the main menu of the program to select the operation to be carried out. For the menu part we define a function ***main_menu*** that displays the option menu and returns the user selection to the main program. This will be the first step to be done in the program loop

```
~/project
Main Menu

Select the desired functionality by typing the corresponding number and Enter
1. Load students list file
2. Load courses file
3. Enroll from file
4. Show course details
5. Show Student details
6. Create course
7. Add student to database
8. Enroll student to course
9. Withdraw student from course
10. Update student grade
11. Save changes
12. Exit
```

At this point the program is waiting for user selection to be input, the ***main_menu*** function have to check for appropriate selection, meaning if user inputs 13 for example it should the main menu back and wait for a proper 1-12 input.

Upon entering a proper selection, the ***main_menu*** function will break out of its loop and return the selection back to the main program where conditional statements will be used to check which option is selected and carry on the corresponding functionality. Remember that we want to always clear the screen and only show the relative information

Important sample screens from some selections

- Option 1, 2, and 3:

The output for option one will reflect the number of students successfully loaded. If we provide the same file in the args to option 1 for example we should get zero students added

```
~/project
Input students list file name (include extension)
students.txt
0 students added to database from file
<<<<<< Press Enter to continue >>>>>>
```

Since we also check for file connection, we should show an error message if the file does not exist

~/project

```
Input students list file name (include extension)
sdfs.txt
Error loading file
0 students added to database from file
<<<<<< Press Enter to continue >>>>>>
```

Option 2 and option 3 have the same operation where they show number of successful loads or failure message if file is not opened successfully

NOTE: for option 3, the provided enrollment files will have the following format as shown in the enrollment sampel file:

- Enroll.txt:

Student_id number_of_enrolled_courses course_id_1 course_id_2 ……. course_id_n

- Option 4:

Calls the function ***show_course_details*** which checks for course id to be in database using the ***find_course*** function. If the course id is not found in the vector of courses we display the following error message

~/project

```
Input ID of course to show:
sdfsdf
Course is not in database
<<<<<< Press Enter to continue >>>>>>
```

If the course is found, the function will use the pointer returned from ***find_course*** to access the course information formatted string function and will show the following screen

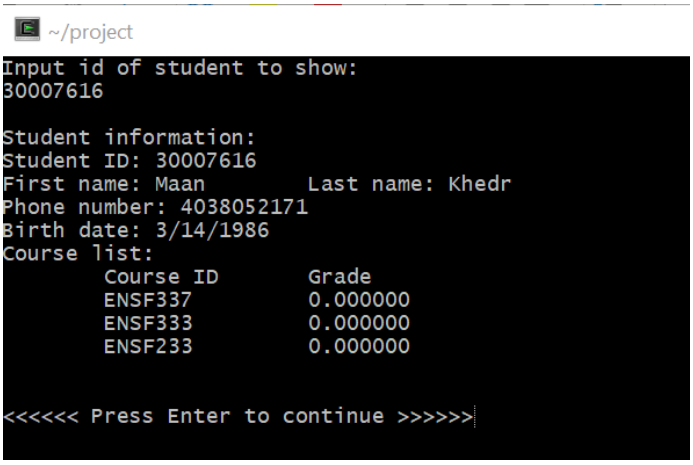
~/project

```
Input ID of course to show:
ENSF337

Course information:
Course ID: ENSF337
Start date: 5/5/2021
End date: 6/17/2021
Capacity: 70
Enrolled: 2
List of students:
        Student name    Grade
        Maan Khedr      0.000000
        diana mcdonald  0.000000


<<<<<< Press Enter to continue >>>>>>
```

Note that the list of students is populated as we loaded an enrollment file beforehand if no enrollment files were loaded the list would be empty

- Option 5:

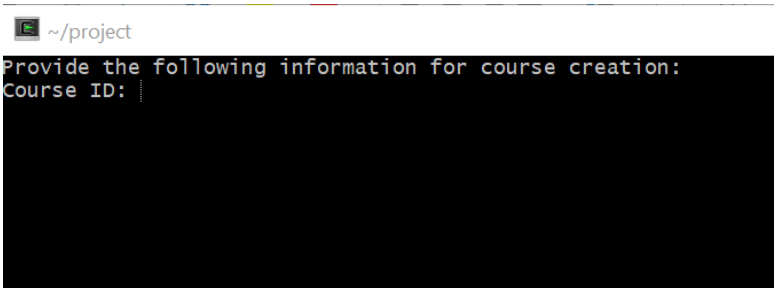
Works exactly the same as option 4 but looks up the student in the student vector instead and shows the appropriate output

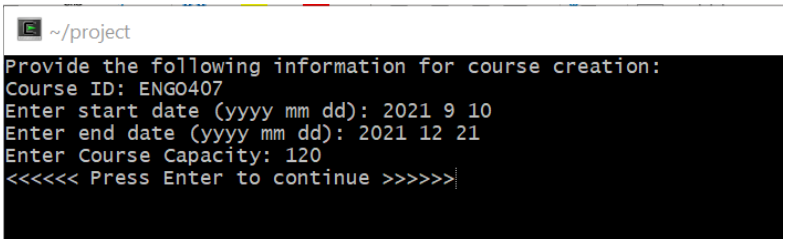


Note that an error message should be displayed if the student is not found in the vector of students.

- Option 6 and 7:

Both options will prompt the user for information of the corresponding object being created. These options utilize the functions ***create_course*** and ***create_student***. These functions will create the object using the information provided by the user from the prompts and will push-back the objects to their corresponding vectors …. Don't forget to check if the object already exists in the vector before you push to avoid duplicates.





.

.

.

.

.

You get the idea by now. Follow the same formatting for output screens for the remaining outputs

Code guidelines:

For your code follow the following guidelines:

1- Use meaningful names and initials for your variables to increase readability.
2- Comment your code:
    a. You don't need to comment every single line of code, instead provide description of code segments/blocks. For example, prior to a loop add a line of two explaining what this loop will do and when it stops in words not equations
3- Follow the guidelines for file error checking as per our lecture notes
4- Don't forget to flush the input stream buffer after using cin to clear any leftovers. That is why there is a function in ***program_functions.h*** named ***buffer_cleaner***
5- You can and will need to declare some variables in your main and functions that are local and were not mentioned in the description. You will need them for evaluating expressions and storing values temporarily from user input to create objects of the classes we defined.
6- Follow the file structure provided by the header files. You will need a corresponding cpp file for each of them in addition to a main.cpp file
7- Understand the code thoroughly, even parts that your teammates developed. You should be able to answer any question during the demo fro any part of the code

Submission guidelines:

Your final code will be submitted on D2L in a designated Dropbox. Your submission will be a compressed file containing:

- Student module (student.cpp and .h)
- Course module (course.cpp and .h)
- Date module (date.cpp and .h)
- Program functions (.cpp and .h)
- Main.cpp
- Students.txt file for loading and storing updated student information for option 11
- Course .txt for loading and storing updated course information for option 11
- Enroll.txt for loading student enrollments to courses and storing updates for option 11