CSE2002 PROGRAMMING II
2023 – 2024 SPRING
PROJECT

SHOULD BE PREPARED AND SUBMITTED INDIVIDUALLY

Date : 13 – 20 May 2024

INSTRUCTIONS

- Add comment for each line of your source code. If your source code **does not include comment(s)**, your source code will be evaluated **over 75 points**.
- If your source code is **compiling with error(s)**, your source code will be evaluated **over 75 points**.
- Your source code must include software writing rules, readability etc.
- Add at the beginning of your source and header files a comment line consisting of **your student-id** and **full name**.
- **Plagiarism is strictly forbidden!** Submitted source code should be the result of your **personal work**!
- Getting help from a 3d party or sharing your work is **prohibited**. The submitted work should be exclusively yours, and your answers should be kept confidential, disclosing them is also a **cheating action**.
- There is **no make-up** for the project. **Late submission and submission via e-mail are not allowed.**
- Your project will be evaluated during the lab hours of the week 20-24 May 2024 in which you are registered. Your project cannot be assessed if you do not present it.
- The projects of students who do not attend laboratory sessions will not be assessed. Late entries will not be accepted. Please come to class on time.

PROJECT REPORT

Perform the following to prepare your **PERSONAL** project report:

- Execute your program as **teacher** and take a screenshot of your output screen. Name it as "**teacher.jpeg**"
- Execute your program as **student** and take a screenshot of your output screen. Name it as "**student.jpeg**"
- Create a word file to explain all functions in detail. Name it as "**functions.docx**"
- Create a folder that contains **your source files, header files**, **word file**, **screenshots**, **teacher office hour file(s)**, and **appointment file,** and name it as "**YourStudentNumber**". Compress the folder by using **winRAR** ("**YourStudentNumber.rar**").

Store the compressed folder ("**YourStudentNumber.rar**") in the "**Project**" folder, located at the course web site under the tab **CSE2002 Programming II /Assignments**.

Program : Office Hour Appointment System

Definition : The program tracks the appointment of office hours.

- The program must contain predefined macros to print file information.
- The program has two users: teacher and student. Prompt to the user to enter "1" for *teacher* and "2" for *student*.

```
The project file information is given below.
      File name:
      Date:
      Time:
      Line:

***************************************************
* WELCOME TO THE OFFICE HOUR APPOINTMENT SYSTEM   *
***************************************************
Who are you? (Enter 1 for TEACHER, 2 for STUDENT):
```

- If the user is a "**teacher**", the program prompts the teacher to enter full name for file name and an operation code, then calls a function to perform the requested action. The codes i (`insert_office_hour`), u (*`update_office_hour`*), p (*`print_office_hour`*), and q (*`quit`*) will be used to represent these operations. Repeats until the teacher enters the command 'q'. Prints an error message if the teacher enters an illegal code.

- If the user is a "**student**", the program prompts the student to enter his/her full name and the teacher's full name. Then, the program loads the office hours of the teacher from the teacher's office hour file and lists the office hours of the teacher by calling *`print_office_hour`* function. The program prompts the student to enter a desired identification number and update the "appointment" file until the student enters the command 'q'.

```
Enter your full name: Student X
Enter teacher's full name: Teacher X
   ID number    Day            Start         End
   1            Monday         1 p.m.        2 p.m.
   2            Wednesday      10 a.m.       11 a.m.

   Enter desired identification number of office hour: 2


Enter teacher's full name: Teacher Y
   ID number    Day            Start         End
   1            Tuesday        3 p.m.        4 p.m.

   Enter desired identification number of office hour: 1


Enter teacher's full name: q
```

- The program will be split into at least four source files:
  - *teacher.c* : functions related to teacher user.
  - *student.c* : functions related to student user.
  - *readLine.c* : functions related to the line.
  - *appointment.c* : contains the main function
- The program will also need header files.

The ***teacher.c*** file will contain the definition for functions that perform the following operations:

- Add new office hours. Prints an error message and returns prematurely if the office hour already exists or the database is full. (**insert_office_hour**) The teacher header file has no prototype for the function store_office_hour that is the helper function to save the office hour in the teacher file. Each teacher has his/her own office hour file named his/her full name.

```
Enter your full name: Teacher X
Enter operation code: i
      Enter identification number: 1
      Enter a day: Monday
      Enter starting hour: 1 p.m.
      Enter ending hour: 2 p.m.
```

- Given an identification number, change the information of the office hour if the office hour already exists; otherwise print an error message (**update_office_hour**). The teacher header file has no prototype for the function find_office_hour that is the helper function to find the office hour in the file.

```
Enter your full name: Teacher X
Enter operation code: u
  Enter identification number: 1
  Enter 1 to update day, 2 to update start, 3 to update end: 1
      Enter new day: Tuesday
```

- Print a table showing all the information in the teacher's office hour file (**print_office_hour**). First, prompt the user to enter his/her name and surname to find related file and list all information.

```
Enter your full name: Teacher X
Enter operation code: p
      ID number   Day         Start       End
      1           Monday      1 p.m.      2 p.m.
      2           Wednesday   10 a.m.     11 a.m.
```

The ***student.c*** file will contain the definition for the function(s) that perform the following operation:

- Create an appointment to store in a file containing the student's name, teacher's name, appointment day, starting hour and ending hour. (**`create_appointment`**)

The **appointment** file of the program should have the following appearance:

```
Teacher      Student      Day          Start        End
Teacher X    Student X    Wednesday    10 a.m.      11 a.m.
Teacher X    Student Y    Monday       1 p.m.       2 p.m.
Teacher Y    Student X    Tuesday      3 p.m.       4 p.m.
```

The ***readLine.c*** file will contain the definition for a function that reads line (`read_line`).

- The user presses the "Enter" key after entering the numerical input, leaving an invisible new-line character that the program must read. When *`scanf`* reads the numerical input, it consumes the 5, 2, and 8, but leaves the new-line character unread. If we try to read the string, it will encounter the new-line character immediately and stop reading. This problem is common when numerical input is followed by character input. One solution is to write a version of ***read_line*** function that skips white-space characters before it begins storing characters. This solves the new line problem and also allows us to avoid storing blanks that precede the character input.