EECE 1080C / Programming for ECE

Summer 2019

Laboratory 6: C++ Structures

Plagiarism will not be tolerated:

all students who share files will receive a 100% penalty on this assignment

Topics covered:

- Structures.
- Classes.

Objective:

- To become familiar with the utility of structure and class type objects in C++. The student will create two short programming assignments to build a better understanding. A third program will expand on the concepts of a previously lab to add increased playability to the program.
- The main goals of this laboratory are as follows:
  - Gain hands on experience with creation and usage of objects in C++.
  - Learn aspects of the language that are uniquely or nearly exclusively associated with objects. These include:
    - Private and public settings.
    - Modifying members.
    - Information exchange in functions.

Collaboration:

- As with most laboratory assignments in this course this laboratory assignment is to be performed by an individual student. You can help each other learn by reviewing assignment materials, describing to each other how you are approaching the problem, and helping each other with syntax errors. You can also get help from teaching assistants and instructors. Having slightly similar code for some assignments is expected but most assignments have multiple different solutions. Your code is expected to be different.
- Please document any help you receive from teaching assistants or instructors. Just add the names to the top of your source file.
- Having someone else code for you, sharing code with other students, or copy-pasting code from the internet or previous terms, **is cheating**. A helper should "teach you to fish, not feed you the fish". Laboratory assignments prepare you for exams so be smart.

Highlights:

- Please consider the following:
  - Review structures.
  - Review classes.
  - Avoid using magic numbers in these functions. Use constants when appropriate.
  - Add a main comment section at the beginning of each program you submit, specifying your name in Pinyin, the date it was last updated and a brief description of what it does.
  - Read problem statements carefully in order to fulfill all instructed requirements.
  - Remember to validate inputs accordingly. It is also a good programming practice.
  - Run your program for all probable outcomes to confirm its functionality. If possible, have other people test your code. This can help identify and troubleshoot problems.
  - Comment important sections of your code to easily recognize your logic and approach to solving the tasks.
- To receive full credit for this laboratory please sign the attendance sheet.
- Please access the laboratory assignment via the canopy/blackboard link. The descriptions for each problem are contained within this document.
- Each part should be worked on separately. You will need a separate project for each part of this assignment when working within your IDE.
- **Submit your .cpp file on Blackboard using the assignment dropbox.**

Rubric: 100 points

- Part A = 30
- Part B = 30
- Part C = 40

Tasks:

A. Time Conversion

- Create a program that will get an integer from the user. This integer represents a total time measured in seconds. Your program needs to output the time in the following standard notation **(hh:mm:ss).**
- Your program **must** include a **struct** that contains:
  - 3 public **integers:**
    - Hours.
    - Minutes.
    - Seconds.
  - 2 public constructors:
    - A default constructor that sets all variables to zero.
    - A user constructor that reads in one variable (total seconds from the user) and separates the total into parts.
  - 2 public functions:
    - An **int** function that converts the standard notation into the total time measured in seconds.
    - A **void** function that prints the total time in the standard notation.
  - A **function** that returns a variable of the **struct type** defined above using the value from the user input divided by 2 (the half time).
- Demonstrate the flexibility of the code by showing the output of three variables in the main program:
  1. A variable of the struct type defined above that uses the default constructor.
  2. A variable of the struct type defined above that uses the user constructor.
  3. A variable of the struct type defined above that uses the separate function.
  4. Using variable #3, output the result of the struct's function that converts the standard notation into the total time.

- **Example 1:**

```
Time (sec): 12345
                                    Using the default constructor: 0:0:0
                                     Using the user constructor: 3:25:45
                                   Using the half-time function: 1:42:52
                                       The half-time total seconds: 6172
```

- **Example 2:**

```
Time (sec): 7891
                                    Using the default constructor: 0:0:0
                                     Using the user constructor: 2:11:31
                                    Using the half-time function: 1:5:45
                                       The half-time total seconds: 3945
```

B. Restaurant Rating

- Create a program that will output a list of your top five favorite restaurants with your personal restaurant rating.
- Your program **must** include a **class** that contains:
  - 2 private variables:
    - A string to store the restaurant name.
    - An integer to store the restaurant rating.
  - 1 public default constructor.
  - 3 public **void** functions:
    - A function to change the set the name of the restaurant using input from the main program.
    - A function to set the rating of the restaurant using input from the main program.
    - A function to print the restaurant name and rating on the same line.
- **Demonstrate the flexibility of the code by printing the output of a vector with 3 restaurants, (you can choose how these will be in the code: user input, written, etc.)**
  1. Use the public functions to set the private data (restaurant name and rating)
  2. Output a list of the restaurants and the rating using the public function to print.

- **Example:**

(optional)

```
These are the three restaurants that Alex rated:
McDonalds: 2
Burger King: 3
KFC: 5
```

C. Game of chance – Multiplayer

- Add a multiplayer functionality to your Game of Chance program from Lab 5.
- Your program must include a struct containing:
  - Player's name.
  - Account balance.
  - Number of wins.
  - Number of losses.
  - Number of games played.
  - (bool) Still in the game.
- Your code should begin by asking Player 1 for the number of players as well as for the starting balance that will apply to all players. Remember to validate both conditions.
- Create a vector using the structure. Use this vector to store all values related to the task.
- Start by prompting all players' names. Consider implementing a **for loop.**
- The game should play out as follows:
  - Each player carries on the game as usual. The player wages and selects a point value. The dice are rolled and the player either wins or loses. Afterwards, the player's turn is over.
  - During gameplay, if a player goes bankrupt (loses all his/her money) or decides to quit (enters 0 wage), that player is immediately removed from the game and the game continues for the remaining players.
  - The game ends when the last player goes bankrupt or folds. At this point, your program should output the statistics of the game for each players. Consider creating a function to accomplish this objective.
- *Programming is all about creativity as well. Remember this is a game so try giving your program a unique game-play feature. For example, consider any of the following*
  - *include fun output messages*
  - *add additional struct variables to track money earned or lost,*
  - *display the number of rounds played,*
  - *display useful information to the user such as the names and balances of remaining players per round*
- *Challenge yourself: adding other elements will help you develop greater programming skills and a better understanding of the usefulness of OOP.*