COMP 3522 Lab 7

Object Oriented Programming in C++ Due

Friday 11:59pm

1 Instructions

This week we will be focussing on operator overloading, using STL containers, iterators and algorithms. You will create a course scheduling program that reads a text file containing courses and times. The program will then print out the course schedules sorted based on day and time along with any time conflicts that occur. Two classes have a conflict if they occur on the same day and their times overlap.

2 Set up your lab

Start by creating a new project:

1. Clone your repo using github classroom: https://classroom.github.com/a/zFrDDnyH
2. Fill out your **name** and **student number** at the top of **main.cpp**
3. Ensure you commit and push your work frequently. You will not earn full marks if you don't

3 Requirements

Remember to create a separate source (.cpp) and header (.hpp) file for each class, plus an additional source (.cpp) file that contains the main method. Search for TODO tags to determine requirements in the code. Implement TODOs wherever you like in code, they do not have to be implemented where the TODO tag appears.

1. Course.cpp is used to implement the following
   (a) Course.cpp represents a course in the schedule
   (b) It contains data that represents the courses' name, day, start and end times
   (c) You will need to implement the overloaded operators =, ==, <, and <<
2. main.cpp is used to implement the following:
   (a) Read from courses.txt
   (b) Store entries from courses.txt into your Course class, and then into an STL container
   (c) Sort your STL container using an **STL sorting algorithm**. Courses occurring on an earlier day and time must appear earlier in the list
   (d) Implement the printSchedule function using an **STL copy algorithm**
   (e) Print any conflicts that occur with courses. Conflicts occur if 2 course have days and times that overlap
   (f) Represent days of the week with a single character: Monday (M), Tuesday (T), Wednesday (W), **Thursday (R)**, Friday (F). There are no classes on Saturday and Sunday

(g) Finally, print the sorted courses to the screen. Given the default values in courses.txt, your output should appear as follows:

```
CONFLICT:
COMP3960 T  1200 1430
COMP3111 T  1200 1430

CONFLICT:
COMP3960 T  1200 1430
COMP3522 T  1230 1430

CONFLICT:
COMP3111 T  1200 1430
COMP3522 T  1230 1430

COMP3222 M  1300 1430
COMP3111 T  1000 1130

COMP3960  T 1200 1430
COMP3111  T 1200 1430
COMP3522  T 1230 1430
COMP3444  T 1600 1730
COMP3333  W 1300 1430
COMP3522  W 1730 2030
COMP3111  F 1000 1130
COMP3960  F 1300 1430
COMP3960  F 1600 1730
```

3. Add or modify code you need to achieve the expected output.
4. Add additional courses into courses.txt to test your code works with other days and times
5. Ensure all member variables are private or protected. Do not use global or public members to store instance data.
6. Ensure you are not duplicating code.

4 Grading

This lab will be marked out of 10. For full marks this week, you must:

1. (2 points) Commit and push to GitHub after each non-trivial change to your code
2. (3 points) Successfully implement the requirements exactly as described in this document
3. (3 points) Successfully test your code.
4. (2 points) Write code that is consistently commented and formatted correctly using good variable names, efficient design choices, atomic functions, constants instead of magic numbers, etc.