ECE 2036 Lab 2: "Project Planning – Gantt Charts"

Due: Sept 23, 2019 @ 11:59PM
Reading: Deitel & Deitel Chapter 2-5

In this lab you will develop a basic software system to generate a Gantt Chart automatically that you can view in a common web browser. You will have as input to your program a text file that contains a description of your project, your team, project duration, and a description of each task. An example input file (called `input.txt`) is given to you in the tarfile (or tarball) that is on Canvas and it also appears in Appendix A. You should be able to easily generalize the input format for any project you choose to create. Your program will read in this file and generate an HTML script (i.e. file) that will display a Gantt chart in a web browser that will help to visual the timeline for project competition. You should load and study the file `exampleGantt.html` into a browser so that you can understand the goal of this project.

The learning objectives of this lab are to give you practice:

1. Using basic C++ classes;
2. Creating basic vectors of user-defined objects;
3. Creating constructors and overloading constructors;
4. Using and creating set and get member functions in C++ classes;
5. Using C++ string objects; and
6. Using basic text file I/O objects.

In the previous lab, you had to create a tarfile (a.k.a. tarball) of your directory for turn in. You will need to do the same thing for your final submission that will look like:

```
tar -cvzf yourusernameLab2.tar.gz ./Lab2
```

However, there are files that you need to get this lab working that I need to provide to you. I have provided skeleton code to help you in your lab creation. You can get a tarball of all the files you need from the Canvas assignment of Lab2. You will upload this tarfile to `coc-ice.pace.gatech.edu` and unpack the files you need using the following command in your HOME directory

```
tar -xzvf lab2codeprovided.tar.gz
```

This should create a Lab2 folder in your main directory that you can now work from.

User Interface and Input Requirements

The user interface for this system is twofold. First the input file that describes the project is specified using specific fields as seen in Appendix A. Second, you will run your program in the following way at the command line.

```
./createGantt inputfile.txt gantChart.html
```

Notice the first string will be the name of your executable file. The second string (`inputfile.txt`) will be the name of the INPUT file that you have created as specified in Appendix A. The third string (`gantChart.html`) is the name of the output file that you would like to create. This output file contains the html commands that can be loaded into your web browser for viewing.

Output Requirements

To generate the output file, you will need to know a little html. I will cover some of this in class; however, I would encourage you to use google to find many tutorials on the subject.

Programming and Testing Requirements

1. I will provide you some skeleton code for your source files and header files. The files are `gantt.h`, `gantt.cc`, and `main.cc`. I have put comments throughout these files to indicate where you need to add code. You will see other files from the unpacked tarfile, please make sure that you leave them in the directory with your source code.
2. I will ask you to use a basic vector template to dynamically create an array of team names and tasks. I will also go over this in class in the slides for this lab. Please note that this is just a brief introduction to vector templates, and we will revisit this later in the semester in greater detailed.
3. In the sample code for the header file that I provide to you, I put the `Task` data in a `struct` complex data type; however, I would like for you to change this to a C++ `class` called `Task`. Make sure that you encapsulate the data and provide a full set of getters and setters. You will also need to change part of the skeleton code throughout the `gantt.cc` file as well so be on the look out!
4. To compile your files you will need to use the following command.

   ```
   g++ gantt.cc main.cc -o createGant
   ```

   This will allow you to run your code as follows

   ```
   ./createGantt inputfile.txt gantChart.html
   ```

5. Once you create your html file, you can test the file on a web browser. Under the "File" menu on your web browser choose the "Open File…" submenu. Find your html file on your local machine, and you should be able to open it.
6. Please create two other test cases and leave in the directory that you turn in. One with a project duration of 1 week and one with a duration of 3 weeks. Call each one of these input files:

   ```
   inputtest1.txt
   inputtest2.txt
   ```

   Please call your output test files

   ```
   ganttChartTest1.html
   ganttChartTest2.html
   ```

   Again, to run your program, I assume that you will use the following command at the command prompt.

   ```
   ./createGantt inputtest1.txt ganttChartTest1.html
   ./createGantt inputtest2.txt ganttChartTest2.htm
   ```

APPENDIX A: Example Input File

```
Project_Name: ECE2030_project
Project_Begin_Date: 09/09/2019
Total_Duration_Weeks: 2
Number_Team_Members: 5
Team_Member_1: Ada_Lovelace
Team_Member_2: Charles_Babbage
Team_Member_3: Grace_Hopper
Team_Member_4: Steve_Wozniak
Team_Member_5: Bill_Gates

Number_of_Tasks: 7

Task_Num: 1
Task_Leader: Bill_Gates
Task_Begin_on_Day: 1
Task_Duration_Days: 3
Task_Description:  Read in data from input file to populate raw data in objects

Task_Num: 2
Task_Leader: Charles_Babbage
Task_Begin_on_Day: 3
Task_Duration_Days: 2
Task_Description: Create basic html file output (with Header Only)

Task_Num: 3
Task_Leader: Grace_Hopper
Task_Begin_on_Day: 5
Task_Duration_Days: 2
Task_Description: Create basic task table (easier part)

Task_Num: 4
Task_Leader: Ada_Lovelace
Task_Begin_on_Day: 6
Task_Duration_Days: 4
Task_Description: Create Gantt Chart with single element to check model

Task_Num: 5
Task_Leader: Steve_Wozniak
Task_Begin_on_Day: 9
Task_Duration_Days: 3
Task_Description: Create Gantt Chart with multiple task

Task_Num: 6
Task_Leader: Grace_Hopper
Task_Begin_on_Day: 12
Task_Duration_Days: 2
Task_Description: Final System Test - 1 day, 6 weeks, 12 weeks

Task_Num: 7
Task_Leader: Bill_Gates
Task_Begin_on_Day: 14
Task_Duration_Days: 1
Task_Description: Ship it! (i.e. upload code to canvas)
```

APPENDIX B: ECE 2036 Lab Grading Rubric

If a student's program runs correctly and produces the desired output, the student has the potential to get a 100 on his or her lab; however, TA's will **randomly** look through this set of "perfect-output" programs to look for other elements of meeting the lab requirements. The table below shows typical deductions that could occur.

In addition, if a student's code does not compile on `coc-ice.pace.gatech.edu`, then he or she will have an automatic 30% deduction on the lab. Code that compiles but does not match the sample output can incur a deduction from 10% to 30% depending on how poorly the output matches the output specified by the lab. This is in addition to the other deductions listed below or due to the student not attempting the entire assignment.

AUTOMATIC GRADING POINT DEDUCTIONS

Element	Percentage Deduction	Details
Does Not Compile on PACE-ICE System	30%	Program does not compile on pace ice cluster!
Does Not Match Output	10%-30%	The program compiles but doesn't match output. For this lab, output will be view from the html file in a web browser.

ADDITIONAL GRADING POINT DEDUCTIONS FOR RANDOMLY SELECTED PROGRAMS

Element	Percentage Deduction	Details
Correct file structure	10%	Does not use both .cc and .h files, implementing class prototype correctly
Does not change Task from a struct to a class	10%	
Encapsulation	10%	Does not use correct encapsulation in object-oriented objects
Setters/Getters	10%	Does not use setters and getters for each data member in Task Class.
Constructors	10%	Does not implement constructors with the correct functionality.
Clear Self-Documenting Coding Styles	5%-15%	This can include incorrect indentation, using unclear variable names, unclear comments, or compiling with warnings. (See Appendix D)

LATE POLICY

Element	Percentage Deduction	Details
Late Deduction Function	score - (20/24)*H	H = number of hours (ceiling function) passed deadline note : Sat/Sun count has one day; therefore H = 0.5*Hweekend