Software Engineering Project 1

(Comp 10050)

Assignment 2 – Implementing a Gantt Chart generator on a command line with a text interface

Deadline 6 pm MARCH 15th 2023

Aim: to create a program that generates a Gantt Chart

You are expected to perform this assignment in a group with just one fellow student who has been randomly assigned with you.

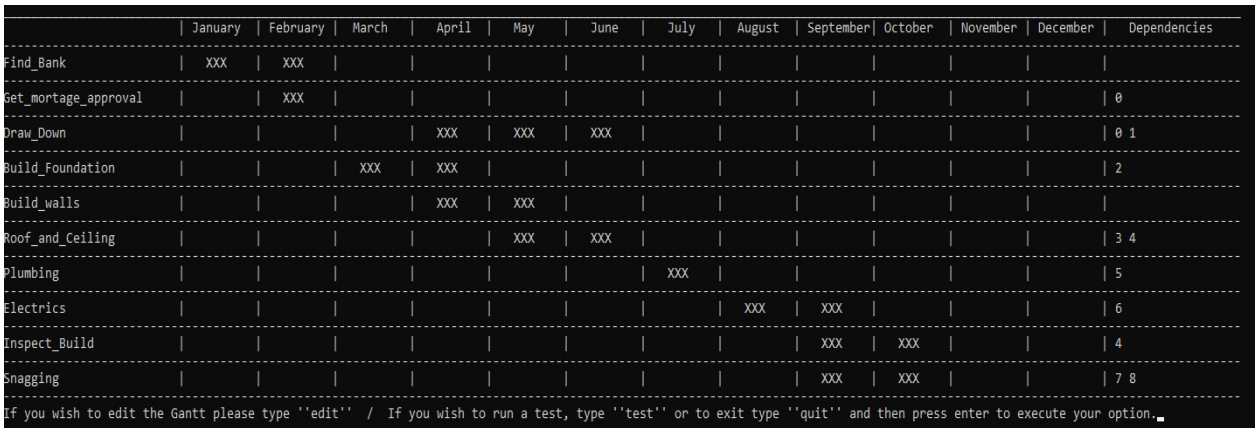
A. Detailed Specification

The objective for this assignment is to create a program that:

1) Asks the user to enter a new series of tasks up to 10 for a Gantt chart or display a predefined Gantt chart with 10 tasks.

For example, this is how the console should ask the user to make this decision. If yes, here is an example of the Gantt chart that was pre-generated. This is only an example and cannot be used, so students need to come up with their series of tasks for the pre-generated option. It is up to the two students to decide how to present the information, but here is an example to base their work on. You can assume, though, that the terminal can be widened to at least a 1920 x 1080 screen for alignment issues.

Welcome to the Gantt Generator
Would you like to use the test example or create your on Gantt from scratch ? (yes or no)
Yes



	January	February	March	April	May	June	July	August	September	October	November	December	Dependencies
Find_Bank	XXX	XXX											
Get_mortage_approval		XXX											0
Draw_Down				XXX	XXX	XXX							0 1
Build_Foundation			XXX	XXX									2
Build_walls				XXX	XXX								
Roof_and_Ceiling					XXX	XXX							3 4
Plumbing							XXX						5
Electrics								XXX	XXX				6
Inspect_Build									XXX	XXX			4
Snagging									XXX	XXX			7 8

If you wish to edit the Gantt please type ''edit'' / If you wish to run a test, type ''test'' or to exit type ''quit'' and then press enter to execute your option.

No

How many tasks would you like to add ? (1-10)
3
Please enter the task name
Task_1
Start month (1-12):
1
End month (1-12):
6
Enter how many dependencies this task has
0
Please enter the task name
Task_2
Start month (1-12):
5
End month (1-12):
8
Enter how many dependencies this task has
1
Enter dependent task
1
Please enter the task name
Task_3
Start month (1-12):
7
End month (1-12):
11
Enter how many dependencies this task has
2
Enter dependent task
1
Enter dependent task
2

2) Then the program should prompt to give them options to edit the tasks, pick a task to check it has circular dependencies as well as print the sequence of its dependant tasks, including those task dependencies' own dependencies in a recursive manner

If you wish to edit the Gantt please type "edit" / If you wish to run a test, type "test" or to exit, type "quit" and then press enter to execute your option.

3) Editing tasks should allow you to change any task by its name. You should then prompt the user for the start and end date. You then must figure out how to take in how many dependencies are necessary for that task and what they are.

If you wish to edit the Gantt please type "edit" / If you wish to run a test, type "test" or to exit, type "quit" and then press enter to execute your option.
edit

Please enter the task name you which to change exactly
Task_1
Please enter the new task name or write its old one
Task_1
Start month (1-12):
2
End month (1-12):
7
Enter how many dependencies this task has
0

4) Implement a test to show if the critical path is possible and list the path e.g. that it is not circular and is a paradox that cannot be completed.

If you wish to edit the Gantt please type "edit" / If you wish to run a test, type "test" or to exit, type "quit" and then press enter to execute your option.
test

10 -> 8 -> 7 -> 6 -> 4 -> 3 -> 1 -> 2 -> (!!!!!!!!!! warning potential circular dependency !!!!!!!!!!!!!)
This warning can then be immediately checked to see if it's real outputting this result if its found.
!!! Circular Dependency Found !!!

Requirements:

1. In writing the program the students are require to upload their progress to a git repository. They must be use https://csgitlab.ucd.ie/ **Week 6 Notes .**

2. The student should add “Abey-Campbell” / abey.campbell@ucd.ie to their cs gitlab repository so we can check their Git commitment when grading to see the work was fairly distributed. If the student pair program then that should be listed in the comments.
3. To create the critical path, the students should use a recursive function **Week 7 notes**
4. The tasks must be a struct array **Week 5 notes**
5. The test data should use the enum’s **Week 5 notes**
6. Clear the screen after you display each Gantt chart. **Week 7 notes**
7. Support for 10 tasks is the requirement, you can support more, but it will not get you extra marks.

Code Design Requirements:

- Comment your code,
- Use functions where you can but you will not be marked against you if you do not use separate files, but it is recommended to separate your code into independent modules, ideally to help with co-development
- The output does not have to match screenshots but should accomplish the same goal of clearly showing the Gantt diagram.

Design Hints:

1. Start by getting the Gantt displaying!
2. The text does not have to match exactly, just the functionality.
3. Think about what you can add to the Struct for a task that will make your life easier e.g. dependencies and how many of them they are .
4. To make sure to clear the screen between draws using system("cls"); and system("clear"); using a #ifdef _WIN32 to decide between them **Week 7 notes**
5. Remember to use the [width] flag for alignment of text e.g. %-25s 25 blank spaces after or %25s 25 blank spaces before using your printf statements **Week 2 notes**

Google strcpy, strcmp, strncmp, and strstr. Get used to one of the standard online resources, e.g.
https://en.wikibooks.org/wiki/C_Programming/C_Reference/string.h/strcpy

Submission:

- Submit two items through Moodle,
    - an archive file (e.g., .zip or .tar.gz) containing your source module
    - a text file (.doc or .pdf) providing implementation details and comments on the design decisions you have made.
        - How is the Gantt displayed
        - How to map the task to struct

- How did you edit and change tasks
- How did you implement search for a circular dependency
- What's your Ascii art about?
- List Git Repo , use your "group name" + " Assignment 2"

Evaluation Criteria

A mark [0-100] will be give according to the following criteria

- Gantt displayed correctly and has a sample task list to start with **(15 points)**
- Git Repo is listed within the text file **(5 points)**
- The code is well commented and appropriately divided into functions **(10 points)**
- You have created an interface to interact with the Gantt chart . **(15 points)**
- The submitted text file describes your design choices appropriately **(10 points)**
- You can input a new task list for the Gantt to show **(15 points)**
- You can edit a task using its name on the Gantt and show the resulting change. **(10 points)**
- You must write a way to test the Gantt diagrams for dependencies to make sure that no task is impossible. This will be based off it know dependencies, you do not need to take into account the timelines of the project in terms of points **(15 points)**
- Adding a personalised piece of ascii art to the main c file **(5 points)**