Department of Computer Science
Bldg 315
900 Fifth Street
Nanaimo, BC
V9R 5S5
Canada

CSCI 161
Spring 2025
Computer Science II

Lab #7 (Mini-Project)

Date Due: Saturday, March 29, 11:59 pm **Total Marks: 100**

General Instructions

- **This lab assignment is individual work. You may discuss questions and problems with anyone, but the work you hand in for this assignment must be your own work. You must give your document the name I prescribe for each question, usually in the form aNqM, meaning Assignment N, Question M. Put your name and student number at the top of every document you hand in. Failure to follow these conventions may result in a deduction of grades for you.**
- **Assignments must be submitted using git.**
- **Git will not let you submit work after the assignment deadline. It is advisable to hand in each answer that you are happy with as you go. You can always revise and push as many times as you like before the deadline.**
- **Partial credit will be given as appropriate, so hand in all attempts.**

Question 1 (100 points):

Purpose: The final assignment is a mini project designed to allow you to design and develop a larger program independently, rather than following the structured design and implementation approaches provided in the labs for smaller problems.

Degree of Difficulty: Moderate

Write complete C++ programs/modules/sub-modules to demonstrate modularity, hierarchy, abstraction, and generalization to represent a simple model as described below.

This assignment specifications (provided below) detail the required behaviour for the mini project, but you are free to go beyond those requirements and add creative features.

Mini project specifications:

You will design and implement your own classes to represent and manipulate seniority and mentoring lists for a restaurant's staff for this mini project.

The restaurant employs four levels of team members:

1. Restaurant manager
2. Crew lead
3. Crew member
4. Candidate (New/probationary employees)

Your software should maintain lists of each employee level (for example, a candidate list, a crew member list, and so on) where:

1. New employees can be hired at any level, but they must start at the end of that level's list (for example, if you are hired as a new crew member, you must start at the end of the crew member list)
2. Existing employees can quit/be fired, in which case they must be removed from their respective lists
3. Current employees can be promoted or demoted, in which case they must be moved to the end of their new level's list
4. Employees can be assigned a mentor from a higher level

Your software must provide the user with a menu (or layer of menus) that allows the user to repeatedly select one of the following actions until they exit the program:

1. hire a new employee (they should be prompted for the employee's name and the level at which they will be hired)
2. promote/demote an employee (again, prompt for the employee's name and new level)
3. dismiss an employee (prompt for the name and remove them from the system)
4. lookup an employee's level (prompt for the name and display their current level)
5. display all employees at a specific level (prompt for which level and display the corresponding employees)
6. assign a new mentor to an employee (prompting for the names of the mentor and mentee)
7. terminate a mentor/mentee pair
8. lookup mentoring information for a specific employee (prompt for their name and display their mentor and their mentee, if any)
9. display all current mentoring pairs (show all mentor/mentee relationships)
10. exit the program

You must create your own linked-list style list implementation, but you have a lot of leeway in designing it. The only requirement is that it be your own design and implementation (you cannot use the lists or related data structures provided in the C++ STL libraries, Boost libraries, or other third-party code). Use meaningful class names and functions to enhance code readability and maintainability.

What to Hand In

Hand in (Git submit) all solution files, Makefile and a README file explaining your program's structure, functionality, and how to run it.

Note: The program should compile and run without modification on a standard C++ compiler (e.g., g++ on Linux/Mac or MinGW on Windows).