Assignment

[Weightage: 10]

Note:

- Plagiarized tasks will be awarded zero marks, does not matter whether one task was plagiarized or all of them.
- Late submission will not be accepted.
- Write clean code with proper indentation and good coding practices.
- Your program must follow all the requirements mentioned in the description.
- Submit the tasks using proper file names in google classroom.

Books Management System for a Digital Library

(Double Linked List)

You are part of a development team, tasked with building a book management system for a digital library. The system is expected to handle a vast collection of digital books, and it should provide a user-friendly interface with menu-driven options to perform various operations. Each book has attributes such as ISBN number, author, title, genre, and publication date (Make a date class in your program to handle dates).

The system should have a user-friendly menu-driven interface with options for different operations. Users should be able to navigate through the following menu items:

1. **Add a New Book:** Users can input book details (ISBN number, author, title, genre, and publication date), and the book will be added to the linked list. When inserting a new book, maintain the order of the books, i-e books must be sorted according to the publication date. Decide where to insert at the time of insertion.
2. **Search for a book:** Using ISBN no or name.
3. **Display Books:** Simple display the linked list.
4. **Filter Books by Author:** Users can input an author's name, and the system should display all books by that author.
5. **Recommend Related Books:** Input a book name, and provide max 3 recommendations for similar books (hint: display books of the same genre).
6. **Delete a Book:** Using it's ISBN number.

Tower of Hanoi - A Historical Challenge

(Use iterative solution with an array-based stack)

Background: The Tower of Hanoi is a classic puzzle that has its origins in an ancient city of Hanoi in Vietnam. According to an information, there was once a temple in Hanoi with three tall pegs (rods) and 64 gold disks of different sizes. These disks were said to represent the universe, and the monks in the temple were tasked with the challenge of moving the entire stack of disks from one peg to another, following three simple rules:

- Only one disk can be moved at a time.
- A disk can only be placed on top of a larger disk or an empty peg.
- The monks must complete the task as quickly as possible.

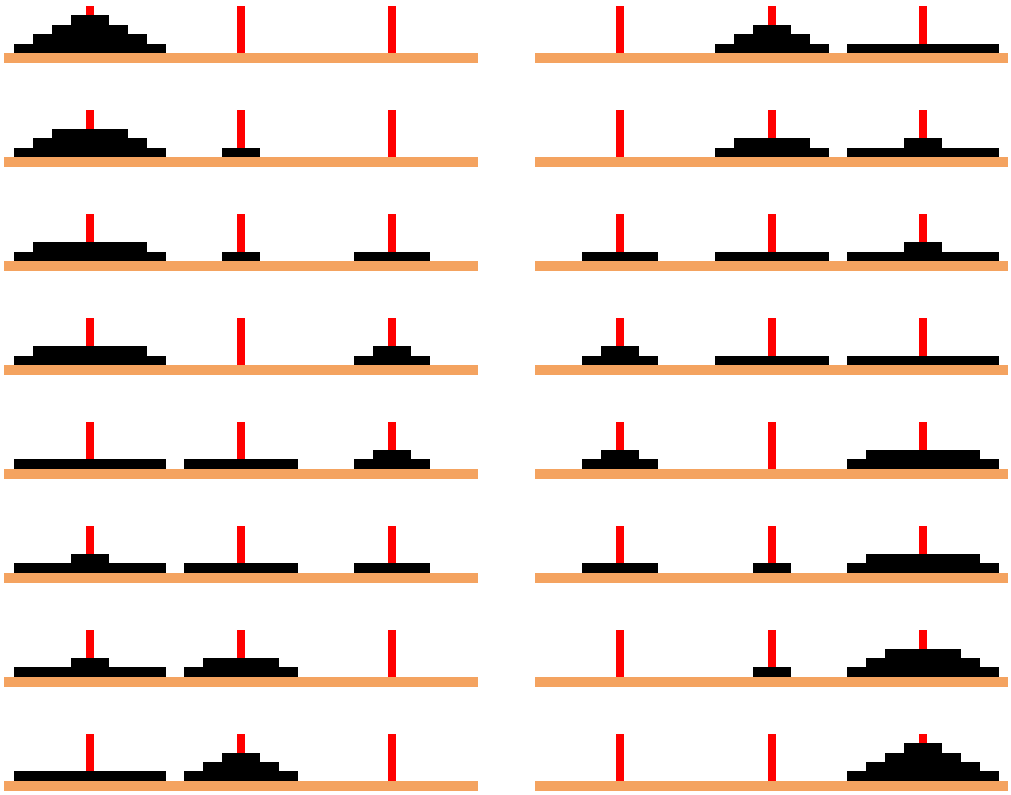
It has been said that the world would end when the monks successfully moved all the disks from one peg to another (just a fictitious story, nothing to do with reality).

Your mission is to implement the Tower of Hanoi puzzle using C++ and the stack data structure. The puzzle will involve moving a specified number of disks from one peg (stack) to another, following the same rules:

- You can only move one disk at a time.
- A larger disk cannot be placed on top of a smaller one.
- You must use an additional peg as an auxiliary rod to accomplish the task efficiently.

Implement a C++ program that can solve the Tower of Hanoi puzzle for a specified number of disks (you can choose the number, e.g., 3, 4, or 5, etc). In your program the numbers will indicate the disk eg. In 3 disks. 1 is the smallest disk, 2 is larger than 1 and 3 is larger than both. When you move disk, show from which rod to which rod which disk was moved in console output.

Your program should display the step-by-step sequence of moves required to solve the puzzle for the chosen number of disks, adhering to the three rules mentioned above. At the end, show that in how many moves, your program placed all of the disk to another stack.



Note: The code for this task is quite simple, so to avoid any plagiarism, submit a detailed dry run on paper (by taking its image). Visualize the memory and output for each step.

University Admission Queue System

(Implement queue using linked list)

You are tasked with designing a queue-based admission system for a FAST university. Each year, a large number of students apply for admission, and the admission process is managed by three admission officers. To ensure efficient processing, students will be directed to the shortest queue for document verification. In your program the student's object should consist of the following info:

- Student name
- Student merit.

So, in your program the Node class should contain the object of student as a data member.

Create a menu-driven program that simulates the university admission queue system. The program should include the following menu options:

1. **Add Student to Queue:** This option allows you to add a new student to the shortest available queue for document verification. In case all the queues has the same number of students, so randomly enqueue in any of the queues.
2. **Process Admissions:** This option begins the admission processing. It will process admissions for each queue in a First-Come-First-Serve (FCFS) manner. Once the shortest queue is empty, the program will automatically start processing the next shortest queue. Make sure your queue is empty after that operation. You will process a student by checking his merit score, if greater than 50, he can secure his admission, else he cannot so just display a message accordingly on screen.
3. **Check Queue Status:** Display the current status of all three queues, including the number of students in each queue and their details.
4. **See who is next:** User can see who is the next student waiting in the specific queue (by entering the queue number).
5. **Exit:** Quit the program.

Organizational Hierarchy Management

(Implement using AVL Tree)

An HR manager in a large corporation, need a system to efficiently manage the hierarchical structure of the organization. The system should store employee information and allow for efficient searching, reporting, and manipulation of the organizational hierarchy.

Design a program with a menu-driven interface that uses an AVL tree to represent the organizational hierarchy of the company. Each node in the tree should contain information about an employee, including their ID, name, and designation.

Menu Options:

Add Employee: This option allows you to add a new employee to the organizational hierarchy. You should input the employee's name, title, department, and any other relevant information. The program should ensure that the tree remains balanced, and employees are added in a way that maintains the hierarchical structure. You should insert the employee at the proper location in the tree based on the employee ID.

1. **Search for Employee:** This option enables you to search for an employee by his ID and the program should display the employee details including his ID, name, and designation. The program should efficiently navigate the tree to find the desired employee and display their information.
2. **Generate Organizational Chart:** This option generates an organizational chart that visualizes the hierarchical structure of the company. The chart should display the names of all employees in a structured manner. You can use the ready-made code to display the tree in a hierarchal manner from internet or from existing codes.
3. **Modify Employee Information:** This option allows you to update the information of an existing employee, such as their name, and designation only (not the employee ID to avoid the tree imbalance).
4. **Remove Employee:** This option lets you remove an employee from the organizational hierarchy. The program should consider various cases, such as the removal of a leaf node, a node with one child, or a node with two children, while maintaining the balanced hierarchy.
5. **Exit:** This option allows you to quit the program.
