DSA LAB(3CS4-21): List of Experiments (Rotar 1)

Experiment 1: WAP to insert an element in the array at the beginning, at the end, and at a specific position. Use a menu-driven approach to define the user-defined function for the given task.

Experiment 2: Write a simple C program on a 32-bit compiler to understand the concept of array storage, the size of a word. The program shall be written illustrating the concept of row major and column-major storage. Find the address of the element and verify it with the theoretical value. The program may be written for arrays up to 4 dimensions.

Experiment 3: Implement Stack using Array: Bookstack Management System
WAP to develop a simple management system for a library that helps track the books returned by patrons. Each book returned should be pushed onto a stack, and you need to implement this functionality using both an array.
Menu for the Program:
Welcome to the Bookstack Management System!
1. Add a book
2. Remove the most recent book
3. View the most recent book
4. Display all books
5. Exit

Experiment 4: Implement Queue using an Array: Customer Service Queue Management System
WAP to develop a simple customer service queue management system for a retail store. The system should manage customers waiting in line to be served. You need to implement this functionality using both an array.
Requirements:

1. **Queue Operations**: Implement the following operations:
   - **Enqueue**: Add a customer to the end of the queue when they arrive.
   - **Dequeue**: Remove the customer at the front of the queue when they are served.
   - **Peek**: View the customer at the front of the queue without removing them.
   - **Display**: Show all customers currently in the queue.
2. **Input Validation**: Ensure the user can’t dequeue from an empty queue or enqueue into a full queue.

Experiment 5: Implement LinkedList:
You are developing a simple task management system for a personal productivity application. The system will manage a list of tasks that a user needs to complete. Each task can be added to the list, marked as completed, or removed from the list. The user should also be able to view all current tasks.
Requirements:

1. **Task Structure: Each task should have the following attributes:**
   - **Task ID (unique identifier)**
   - **Task description (text)**
   - **Status (e.g., "Pending" or "Completed")**
2. **Linked List Operations:**
   - **Add Task: Allow the user to add a new task to the end of the list.**
   - **Remove Task: Allow the user to remove a task by its ID.**
   - **Mark Task as Completed: Allow the user to update a task's status to "Completed."**
   - **Display Tasks: Provide a way for the user to view all current tasks in the list.**

Experiment 6: Represent a 2-variable polynomial using array. Use this representation to implement the addition of polynomials. Also use a header-linked list to manage polynomials stored in an array that can help in efficiently handling dynamic polynomial terms, allowing for easy addition and removal of terms without worrying about fixed array sizes.