Lab Tasks

1. Imagine you are tasked with creating a program to manage a library's book inventory. Each book has attributes such as title, author, publication year, and genre. Design a **struct** that effectively represents a book as a real-world entity. Then, write a C++ program that utilizes this **struct** to demonstrate the creation, modification, and display of book information.
    a) Extend the program to manage multiple books in an array.
    b) Implement a function to search for a book by title or author.
    c) Allow the user to input new books and update existing book information interactively.

2. You are tasked with building a simple product management system for an online store that does the following stuff:
    a) Create a function that allows the addition of a new product to the system. The function should take parameters such as product name, price, quantity, and any other relevant details.
    b) Implement a function that takes a product ID as input and displays detailed information about the product, including its name, price, quantity in stock, and any other relevant details.
    c) Design a function that enables the update of product information. It should take a product ID as well as the new details (e.g., updated price, quantity, etc.) and modify the existing product's information accordingly.
    d) Create a function that removes a product from the system based on its product ID. Ensure that the inventory is updated after the removal.

3. Implement a function that dynamically allocates an array of integers. The function should take an integer parameter specifying the number of elements to allocate and return a pointer to the allocated array.

4. Create a program that moonlights as a basic calculator, wielding the power of pointer arithmetic to crunch numbers like a pro. It should charm the user into entering two integers, perform addition, subtraction, multiplication, or division (depending on the user's mood), and then proudly display the result. Use pointers to juggle values between functions and make the magic happen.

5. You need to make a program that works like a Matrix Wizard. First, it will ask the user for the size of a 2D matrix and then create it using memory magic (dynamic allocation). Next, the program will fill the matrix with numbers given by the user. After that, it will show off by doing matrix addition, subtraction, and multiplication (only if the sizes match). Finally, it will display the results and clean up the memory like a good, tidy wizard.

6. Time to code a program that manages an ever-growing hungry integer array! The array starts small with a size of 5, but every time you keep stuffing it with numbers and it gets full, it will double in size to satisfy its appetite. Once you're done feeding it all the elements, the array will go on a diet and shrink down to perfectly fit the number of elements it holds. No wasted space, no extra fluff—just a happy, well-fed array.