# C++ OOP and Data Structures

Instructions:  
- Answer all questions.  
- Write clear and commented code where necessary.  
- Duration: 2 hours.

A screen shot of a computer screen

Description automatically generated**Q1**. Print the following pattern using nested loops:

A math problem with numbers

Description automatically generated**Q2.** An Armstrong number (or narcissistic number) for a given number of digits is a number such that the sum of its digits raised to the power of the number of digits equals the number itself. For example: (1≤N≤105 )

**Q3**. C++ program to remove duplicate values from a vector

Input:

Vector 1: {1, 2, 2, 3, 4, 4, 5, 6, 6}:

Output:

Vector after removing duplicates: 1 2 3 4 5 6

## Problem 1: Bank Account Management

Create a class BankAccount with:

1. Private attributes: accountNumber (string), balance (double).
2. Methods:
   * deposit(double amount) to add money to the balance.
   * withdraw(double amount) to subtract money from the balance (ensure sufficient funds).
   * displayAccountDetails() to display the account number and balance.

Write a main() function to:

1. Create a BankAccount object.
2. Perform deposit and withdrawal operations.
3. Display the account details.

## Problem 2: Vector Operations in a Class

Create a class `NumberList` with:  
1. A private attribute `numbers` (vector of integers).  
2. A method `addNumber(int n)` to add a number to the vector.  
3. A method `removeOddNumbers()` to remove all odd numbers from the vector.  
4. A method `displayNumbers()` to print all numbers in the vector.  
  
Write a `main()` function to:  
1. Create a `NumberList` object.  
2. Add 10 integers to the vector.  
3. Remove odd numbers and display the remaining numbers.

## Problem 3: Employee Management System

Create a class Employee with:

1. Private attributes: id (int), name (string), salary (double).
2. Methods:
   * A constructor to initialize an employee's details.
   * displayDetails() to print the employee's details.
3. A static method compareSalary(Employee e1, Employee e2) to compare the salaries of two employees and return the one with the higher salary.

Write a main() function to:

1. Create two Employee objects.
2. Display their details.
3. Compare their salaries and display the employee with the higher salary.

## Problem 4: Inheritance and Polymorphism

Create a base class `Shape` with:  
1. A pure virtual method `calculateArea()`.  
  
Create two derived classes:  
1. `Rectangle` with attributes `length` and `breadth`. Implement `calculateArea()` to calculate the rectangle's area.  
2. `Circle` with attribute `radius`. Implement `calculateArea()` to calculate the circle's area.  
  
Write a `main()` function to:  
1. Create objects of `Rectangle` and `Circle`.  
2. Calculate and display their areas.

## Problem 5: Library Management System

Create a class Book with:

1. Private attributes: title (string), author (string), isAvailable (bool).
2. Methods:
   * A constructor to initialize the book's details.
   * borrowBook() to mark the book as borrowed if available.
   * returnBook() to mark the book as available.
   * displayBook() to print the book's details.

Write a main() function to:

1. Create a list of 3 Book objects.
2. Borrow and return books using the appropriate methods.
3. Display the updated book details.

## Problem 6: Linked List Implementation

Create a class LinkedList to implement a singly linked list with the following features:

1. A private structure Node with two attributes: int data and a pointer next to the next node.

2. A private attribute Node\* head to point to the first node.

3. Methods:

* + insertNode(int n) to insert a new node at the end of the list.
  + deleteNode(int n) to delete the first node containing the value n.
  + displayList() to display all the elements in the linked list.

Write a main() function to:

1. Create a LinkedList object.

2. Insert 5 integers into the linked list.

3. Delete a specific node and display the updated linked list.

**Problem 7: Spiral Matrix Traversal**

Create a program to:  
1. Define a 2D array `matrix` of size 4x4.  
2. Traverse and print the elements of the matrix in a spiral order (clockwise from the top-left corner).  
3. Ensure the program works for matrices with arbitrary sizes (NxM) as well.  
  
Write a `main()` function to:  
1. Input a 4x4 matrix.  
2. Print the matrix elements in spiral order.

**Problem 8: Submatrix with Maximum Sum**

Create a program to:  
1. Define a 2D array `matrix` of size 6x6.  
2. Identify the 3x3 submatrix within the larger matrix that has the maximum sum of its elements.  
3. Print the original matrix, the identified submatrix, and the maximum sum.  
  
Write a `main()` function to:  
1. Input a 6x6 matrix.  
2. Calculate the submatrix with the maximum sum and display it.

**Problem 9: Row and Column Swapping**

Create a class `MatrixOperations` with:  
1. A private attribute `int mat[4][4]`.  
2. Methods:  
 - `setMatrix()` to input the elements of the matrix.  
 - `swapRows(int row1, int row2)` to swap two rows of the matrix.  
 - `swapColumns(int col1, int col2)` to swap two columns of the matrix.  
 - `displayMatrix()` to display the matrix.  
  
Write a `main()` function to:  
1. Create a `MatrixOperations` object.  
2. Input a 4x4 matrix, perform row and column swaps as specified by the user, and display the updated matrix.

## Problem 10: Class with Array and Loop

Create a class `Student` with:  
1. A private attribute `marks` (array of 5 integers).  
2. A method `setMarks()` to take input for the 5 marks.  
3. A method `calculateAverage()` to calculate and return the average of the marks using a loop.  
4. A method `displayMarks()` to print all 5 marks.  
  
Write a `main()` function to create a `Student` object, input marks, calculate the average, and display the marks and average.

## Problem 11: Queue Operations in a Class

Create a class QueueOperations with:

1. A private attribute queue<int> data.
2. Methods:
   * enqueue(int n) to add an element to the queue.
   * dequeue() to remove the front element.
   * displayQueue() to print all elements in the queue.

Write a main() function to:

1. Enqueue 5 integers into the queue.
2. Dequeue the first 2 elements.
3. Display the remaining elements.

## Problem 12: Stack and Encapsulation

Create a class `StackOperations` with:  
1. A private attribute `stack<int> data`.  
2. Methods to perform the following stack operations:  
 - `push(int n)` to add an element to the stack.  
 - `pop()` to remove the top element.  
 - `displayStack()` to print all elements of the stack (use a loop).  
  
Write a `main()` function to demonstrate:  
1. Pushing 5 integers onto the stack.  
2. Popping the top 2 elements.  
3. Displaying the remaining elements.

## Problem 13: Matrix Operations

Create a class Matrix with:

1. A private attribute vector<vector<int>> mat to store a 2D matrix.
2. Methods:
   * setMatrix(int rows, int cols) to take input for a matrix of given dimensions.
   * transpose() to calculate and return the transpose of the matrix.
   * displayMatrix() to print the matrix.

Write a main() function to:

1. Create a Matrix object.
2. Input a 2D matrix.
3. Display the original and transposed matrices.

## Problem 14: Sorting and Searching in OOP

Create a class `DataProcessor` with:  
1. A private attribute `vector<int> data`.  
2. A method `addData(int n)` to add elements to the vector.  
3. A method `sortData()` to sort the vector in ascending order using the `sort()` function.  
4. A method `searchData(int n)` to perform a linear search for an element in the vector.  
  
Write a `main()` function to:  
1. Add 10 integers to the vector.  
2. Sort the vector and display the sorted elements.  
3. Search for a specific number and display whether it is found or not.

## Problem 15: Shopping Cart System

Create a class ShoppingCart with:

1. A private attribute map<string, int> items where the key is the item name, and the value is the quantity.

2. Methods:

* + addItem(string itemName, int quantity) to add or update the quantity of an item in the cart.
  + removeItem(string itemName) to remove an item from the cart if it exists.
  + displayCart() to display all items and their quantities in the cart.
  + Write a main() function to:

1. Add 5 items to the shopping cart.

2. Remove 1 item from the cart.

3. Display the updated cart items.