Data Structures and Algorithms — Lab 3

## **Topic**

We will reinforce the concepts of Abstract Classes, Pure Virtual Functions, and Array Manipulation through the implementation of a generic data structures.

## **Objectives**

* Understand the purpose of abstract classes and how they act as a blueprint for derived classes.
* Explore the use of pure virtual functions and how they enforce functionality in derived classes.
* Practice the implementation of polymorphism and inheritance.
* Develop and manipulate custom data structures with advanced functionalities.
* Gain familiarity with practical applications of generics and modular design in C++.

## **Outcomes**

1. Develop a comprehensive understanding of creating abstract classes and using them to enforce specific behaviors in derived classes.
2. Gain experience implementing pure virtual functions to define essential operations within abstract data structures.
3. Build practical skills in implementing a generic list data structure with advanced functionality such as custom insertion, searching, rotation, and prime number identification.
4. Enhance problem-solving skills by designing functions that meet specific operational requirements, such as prime sum calculation and rotation mechanisms.

## **Content**

The following sections will be covered during this lab session:

1. **Implementation of Basic List Operations**
   * Create MyList as a derived class from List with fundamental list functionalities such as insertion, deletion, and searching.
   * Integrate helper functions (empty, full, size) to monitor list status and enhance usability.
2. **Advanced List Manipulation**
   * Extend MyList into CustomList with specialized methods:
     + **Prime Sum Calculation:** Identify and sum all prime numbers within the list.
     + **Second Maximum and Minimum Functions:** Retrieve the second highest even and second lowest odd values.
     + **Duplicate Detection:** Identify and print duplicate elements.
     + **List Rotation:** Perform clockwise and anti-clockwise rotations on split halves of the list.
3. **Practical Applications of Virtual Functions and Polymorphism**
   * Explore polymorphism by implementing function overrides for unique list operations across derived classes.
   * Reinforce the modularity and extensibility of list classes by adding new functionality without altering the base class.

## **Task Instructions**

Students are required to complete the following tasks during lab time.

Create a private repository on your GitHub accounts. The name of the repository should be: **Lab-3-DSA**. All files should be uploaded to this repository; this includes the header and cpp files. Please note that the name of class-related files should be the same as the class name. We can name the main file Task\_1.cpp or Task.cpp if there is only 1 task.

We recommend you work in .h files for classes only since we must work on templates.

## **Task 1**

Create a C++ generic abstract class named **List**, with the following:

**Attributes:**

Type \*arr;

int maxSize;

int currentSize;

**Functions:**

* 1. virtual void addElementAtFirstIndex(Type) = 0;

// Should add the element at the first position of the **List**

* 1. virtual void addElementAtLastIndex(Type) = 0;

// Should add the element at the last position of the **List**

* 1. virtual Type removeElementFromEnd() = 0;

// Should remove the element from the last position of the **List**

* 1. virtual void removeElementFromStart() = 0;

//Should remove the element from the first position of the **List**

* Write a **parameterized constructor with default arguments** for the above class.
* Write a **copy constructor** for the above class.
* Write **destructor** for the above class.

## **Task 2**

Create a menu-based program for the following functions, using the class made in task 1; make a class named as **MyList**, having the following additional functionalities:

* **bool empty()**: Returns whether the MyList is empty or not
* **bool full():** Returns whether the MyList is full or not  
  **int size():** Returns the current size of the MyList
* **bool insertAt(int index, T value):** Adds a value at the index passed to the function, returns true if the index is present and value is added else returns false.   
  **Type last():** Returns the last element of the MyList
* **bool search(Type):** Returns true if the searched value is present in the list else returns false
* Write a **parameterized constructor with default arguments** for the above class.
* Write a **copy constructor** for the above class.
* Write **destructor** for the above class.

## **Task 3**

Create a menu-based program for the following functions, using the class made in task 2, make a class named as **CustomList**, having the following additional functionalities:

* **Type sum\_ofPrime()**: It finds prime numbers in the list and calculates and returns the sum of all prime numbers present in list.
* **Type secondMaxEven()**: It finds and returns the **second maximum even number** present in list.
* **Type secondMinOdd()**: It finds and returns the **second minimum odd number** present in list.
* **void printDuplicates():** It finds and displays all the numbers which occur more than once.
* **void rotateClockwaise( int r)**: It divides the list into two parts (halves the list), and rotate both parts “**r**” times in clockwise direction
* **void rotateanitclockwaise( int rt)** : It divides the list into two parts (halves the list), and rotate both parts “**rt**” times in the anti-clockwise direction
* Write a **parameterized constructor with default arguments** for the above class.
* Write a **copy constructor** for the above class.
* Write **destructor** for the above class.