University of Central Punjab
(Incorporated by Ordinance No. XXIV of 2002 promulgated by Government of the Punjab)
FACULTY OF INFORMATION TECHNOLOGY

Data Structures and Algorithms - Lab

Lab 2

CLO NO	CLO STATEMENT	Blooms Taxonomy Level	PLO
1	Solve real-world problems skillfully with precision using programming constructs learned in theory with the course toolkit.	P3	5

Task 1

Create a function swap() that interchanges the values of the two arguments sent to it. Make the function into a template, so it can be used for all data types (int, float, double, char, etc.). Write a main program to exercise the function with several different types.

Sample Output:

Enter two integers: 5 10
Before swapping (int): x = 5, y = 10
After swapping (int): x = 10, y = 5

Enter two decimal numbers: 3.14 2.71
Before swapping (double): a = 3.14, b = 2.71
After swapping (double): a = 2.71, b = 3.14

Enter two characters: A Z
Before swapping (char): c1 = A, c2 = Z
After swapping (char): c1 = Z, c2 = A

Task 2

Write a template function that returns the average of all the elements of an array. The arguments to the function should be an array name and the size of the array(int). In main function, exercise the function with arrays of type int, long, double and char.

Sample Output:

Enter the number of elements for int array: 5
Enter 5 integer values: 10 20 30 40 50
Average of int array: 30

Enter the number of elements for double array: 4
Enter 4 decimal values: 1.1 2.2 3.3 4.4
Average of double array: 2.75

Enter the number of elements for char array: 3
Enter 3 characters: A B C
Average of char array (based on ASCII values): 66

Task 3

Write a C++ template class Storage that can store a single value of any data type. The class should have the following functionalities:

- A constructor to initialize the value.
- A setValue() function to update the stored value.
- A getValue() function to retrieve the stored value.

Write a main () function to test the Storage class with different data types (e.g., int, double, string).

Note: Do not use inheritance in your implementation.

Task 4

Create a C++ class named StudentRecords that manages an array of student names and their corresponding scores. The class should be implemented using separate header (.h) and implementation (.cpp) files.

Class Requirements:

1. Constructor to initialize the array with a given number of students.
2. Destructor to free dynamically allocated memory.
3. setStudent(index, name, score) – A function to set the student's name and score at a given index.
4. getStudent(index) – A function to retrieve the student's name and score.

5. findTopScorer() – A function that returns the name of the student with the highest score.
6. findLowestScorer() – A function that returns the name of the student with the lowest score.
7. displayAll() – A function to print all student names and scores.

Sample Output:
Enter number of students: 3
Enter details for 3 students:
Student 1 Name: Alice
Student 1 Score: 85
Student 2 Name: Bob
Student 2 Score: 90
Student 3 Name: Charlie
Student 3 Score: 78

Student Records:
Alice - 85
Bob - 90
Charlie - 78

Top Scorer: Bob (90)
Lowest Scorer: Charlie (78)

Instructions:
1. StudentRecords.h – Declare the class and its functions.
2. StudentRecords.cpp – Define the functions.
3. main.cpp – Write a test program to use the class.

Task 5

Create a **C++ template class** named ArrayProcessor that manages a **dynamic array** of any data type. The class should be implemented using **separate header (.h) and implementation (.cpp) files**.

Your class should support the following operations:
1. **Constructor** to initialize the array with a given size.
2. **Destructor** to free dynamically allocated memory.
3. **setElement(index, value)** – A function to set the value at a given index.
4. **getElement(index)** – A function to get the value at a given index.
5. **findMax()** – A function that returns the maximum element in the array.
6. **findMin()** – A function that returns the minimum element in the array.
7. **reverseArray()** – A function to reverse the order of elements in the array.

Sample Output:
Enter size of integer array: 5
Enter 5 integers: 10 25 5 40 15

Original Array: 10 25 5 40 15
Maximum Element: 40
Minimum Element: 5
Reversed Array: 15 40 5 25 10

Enter size of double array: 4
Enter 4 double values: 1.5 3.7 0.8 2.9

Original Array: 1.5 3.7 0.8 2.9
Maximum Element: 3.7
Minimum Element: 0.8
Reversed Array: 2.9 0.8 3.7 1.5

Instructions for Implementation:

- Create a header file ArrayProcessor.h to declare the class.
- Create a separate implementation file ArrayProcessor.cpp to define the functions.
- Write a main.cpp file to test the class with int and double arrays.

Task 6

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

- Type *arr;
- int maxSize;
- int currentSize;

Functions:

- virtual void addElementAtFirstIndex(Type) = 0;
  // Should add the element at the first position of the **List**

- virtual void addElementAtLastIndex(Type) = 0;
  // Should add the element at the last position of the **List**

- virtual Type removeElementFromEnd() = 0;
  // Should remove the element from the last position of the **List**

- virtual void removeElementFromStart() = 0;
  //Should remove the element from the first position of the **List**

Instructions:

- 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 7

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

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

Instructions:

- 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.