National University of Computer and Emerging Sciences

Lab Manual 01
Data Structures

Course Instructor	Miss. Arooj Khalil
Lab Instructor	Miss. Saira Arshad
Section	BCS-3F2
Semester	Fall 2023

Department of Computer Science
FAST-NU, Lahore, Pakistan

Task1

Implement a class called Library that manages a collection of books. The Library class should have the following members:

1. Private data member:
   Char*books
   capacity - an integer representing the maximum number of books the library can hold.
   count - an integer representing the current number of books in the library (initially set to zero).

2. Default constructor - Initialize the capacity to a default value

3. Constructor with an integer parameter

4. Copy constructor - Create a deep copy of another Library object.

5. Destructor - Release any dynamically allocated memory (if needed).

6. Member function: addBook - Takes information about a book as parameters and adds it to the library. This method should add one integer to the array and increase the number of current elements. If the allocated size is not sufficient then increase the capacity of the array by double. Copy the old array in the new one (including the new element) and delete the old array.

7. Member function: removeBook - Takes the ISBN of a book and removes it from the library.

8. Member function: searchBook -
9. Member function: printLibrary - Print the information about all the books in the library.

10. Create a main program to test the Library class. Perform operations like adding books, removing books, searching for books, and printing the library's contents.

Task2

Implement a Complex class to represent complex numbers. Complex numbers have a real part (a) and an imaginary part (b) and are usually written in the form a + bi. The Complex class should have the following members:

1. Private data members real and imaginary to store the real and imaginary parts of the complex number.

2. A constructor that takes two double values as parameters to initialize the real and imaginary parts.

3. Overload the << operator to allow printing complex numbers in the format a + bi.

4. Overload the >> operator to allow input of complex numbers in the format a + bi.

5. Overload the + operator to add two complex numbers together and return a new complex number.

6. Overload the - operator to subtract one complex number from another and return a new complex number.

7. Overload the * operator to multiply two complex numbers together and return a new complex number.

8. Overload the / operator to divide one complex number by another and return a new complex number.

9. Implement a member function abs() to calculate the absolute value (magnitude) of the complex number.
10. Implement a member function conjugate() to calculate the conjugate of the complex number (negate the imaginary part).

11. Implement a member function equals() that checks if two complex numbers are equal.

12. Write a sample program to test your Complex class by performing arithmetic operations, finding the absolute value, calculating the conjugate, and checking for equality between complex numbers.