National University of Computer and Emerging Sciences



Lab Manual 04

Object Oriented Programming

|  |  |
| --- | --- |
| Course Instructor | Mr. Waqas manzoor |
| Lab Instructor (s) | Mr. Dilawar Shabbir |
| Section | BSE-2C |
| Semester | Spring 2022 |

Department of Computer Science

FAST-NU, Lahore, Pakistan

## Objectives

After performing this lab, students shall be able to:

* Copy Constructor
* Shallow vs deep copy constructor
* Dynamically allocate and deallocate memory
* Array of Objects

**Instructions**

* Make sure memory is deallocated properly
* implement the functions outside the class
* Make appropriate functions if you need them.

**Task 01:**

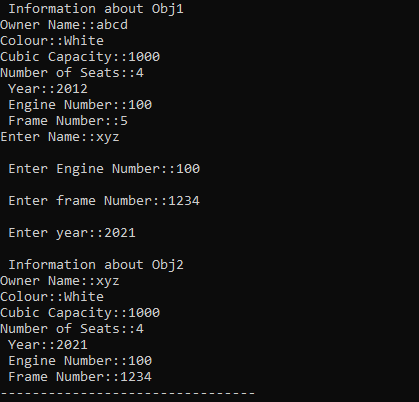
Lamborghini is an international luxury sports car developer stationed in Italy. The company has a reputation for producing cars that are extremely expensive, powerful and rare. Lamborghini has developed a brand new model called the Diablo. The company produces a very limited number of Diablo’s each year. The company is producing the Diablo in only one colour called the “Hot Red”.

When the company has produced a Diablo, the car has a number of attributes like colour, cubic capacity, number of seats, year of manufacture, engine number, frame number and owner name. Out of these attributes the attributes that remain the same for all Diablo’s being produced are colour, cubic capacity and number of seats.

Suppose you are working on a system specially designed for the Lamborghini Diablo. Follow the instructions below for creating the class and objects:

* Create default argument constructor.
* Create an object named “obj1” and initialize the object.
* Create a copy constructor that can copy all those attributes that remain the same for all cars.
* Create getter and setter functions to set and get the data of the data member
* Generate another object named “obj2” that is created by copying only those attributes that are the same from “obj1”.
* Initialize the remaining attributes with values of your own.

**Example input and output:**

****

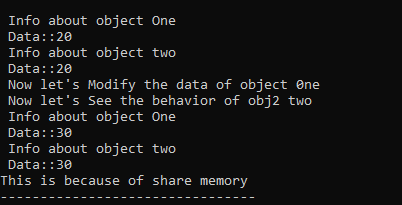
**Task 02:**

Create a class Student

* It has a data member integer array of 5 subjects marks
* Create a default constructor that assigns dynamic memory to the data member
* Make setter function named void set\_marks(int marks,int index) that will receive marks and index location to set the marks(**note**: please validate the index location before setting the marks)
* Create object student1 in the main and set the marks of five subjects by calling set\_marks function through loop.
* Create a function display( ) that will display the data of the objects
* Now create another object student2 that will initialize with the existing object and display the data of both objects
* Now make a destructor that will deallocate the dynamic memory of the data member
* Deallocate the memory of student1 object and now display the data of object student2

Now note the issue and give the reason why it is happening.

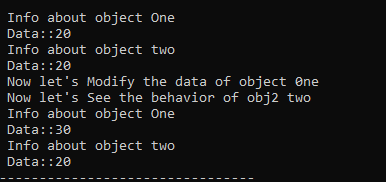
**Sample:**

****

**Task 03:**

Once you have done **practice task 02** and demonstrate the issue. The overload copy constructor and give the implementation of deep copy inside the overloaded copy constructor. Now call the display function to check whether the issue has been resolved or not.

**Example:**



**Task 04:**

Create a dynamic size list in the “driver.cpp” and implement all the functions of class named **LIST** in “list.cpp” (remember you need to define your class in “list.h” file).Use dynamic memory allocation for list.

**1-INSERT**

1. Create a [sorted array](https://www.geeksforgeeks.org/program-check-array-sorted-not-iterative-recursive/) **arr[]** consisting of **N**distinct integers now take an input integer **K from the user**, the task is to find the index of K, if it’s present in the array **arr[]**. Otherwise, find the index where **K** must be inserted to keep the array sorted.
2. Create an array of set of non-overlapping intervals. Now take input of new interval from the user, insert the interval at correct position. If the insertion results in overlapping intervals, then merge the overlapping intervals. Remember that the set of non-overlapping intervals is sorted on the basis of start time, to find correct position of insertion.

**Example:**

**Input:** Set : [1, 3], [6, 9]

New Interval : [2, 5]

**Output:** [1, 5], [6, 9]

The correct position to insert new interval

[2, 5] is between the two given intervals.

The resulting set would have been

[1, 3], [2, 5], [6, 9], but the intervals

[1, 3], [2, 5] are overlapping. So, they are

merged together in one interval [1, 5].

**2-SEARCH**

Create a dynamic size unsorted array of distinct integers and take an element **x from the user**. Search the element **x** in the array using minimum number of comparisons. Any sort of comparison will contribute 1 to the count of comparisons. For example, the condition used to terminate a loop, will also contribute 1 to the count of comparisons each time it gets executed. Expressions like **while (n) {n–;}** also contribute to the count of comparisons as value of **n** is being compared internally so as to decide whether or not to terminate the loop.

**3-DELETE**

**Delete** all the peak elements from the list used in search function (An array element is a peak if it is NOT smaller than its neighbors. For corner elements, we need to consider only one neighbor). **.Delete** all used list with any memory leakage using delete function in LIST class.

**END**