**National University of Computer and Emerging Sciences**



**Lab Manual 05**

**Object Oriented Programming**

|  |  |
| --- | --- |
| Course Instructor | Syeda Tayyaba Bukhari |
| Lab Instructor (s) | Fariha Maqbool  Muhammad Usama |
| Section | BCS-2H |
| Semester | Spring 2023 |

Department of Computer Science

FAST-NU, Lahore, Pakistan

## Objectives

After performing this lab, students shall be able to:

* Understand shallow copy and deep copy
* Create default, parametrised and copy constructor
* Pass objects as parameters to functions

**Task#1:**

Create a class Student with data members (size, an integer variable and marks, a pointer to integer and it should be used to declare an array inside the constructor).

create a Default constructor that should initialize the pointer with NULL and size with 0, and a parameterized constructors that should receive an integer array and size as parameter.

**Note :** Parameterized constructors should receive an array from main function and declare and initialize the array ‘marks’ with the received array i.e., (implement deep copy)

Now Create a function “scale up”. This function should increase the marks of each student by 5.

**Note:** make sure that by adding the value 5 the updated value should not exceed 100 and similarly don't apply this scale up factor on the marks less than 45.

Create a function “printInfo” to print the data.

create a static array Inside main function and initialize with hard coded values. You can also initialize with user input.

Now create the object of class and pass the local array, and its size into the parameters.

Call printInfo function and then Call the function “scale up”.

Call the printInfo function again after calling the scale up function and check whether the data is updated or not and whether it is in the valid range or not.

Create a destructor inside the class that should de-allocate the dynamically allocated memory and initialize the pointer with nullptr to handle dangling pointer.

**Task#2:**

Implement a class called **MyBigInt**. The MyBigInt class will have two data members:

* int\* big\_int\_; // Pointer to the int array that holds the big integer
* int int\_length\_; // Variable to store the length of the big integer

While an integer is of 4 bytes in size with a range of -2,147,483,648 to 2,147,483,647. A big integer can store long integer numbers with no size limitation. (Each Index of big\_int will hold one digit).

You have to implement the following member functions:

1. Write a default constructor and initialize big\_int\_ to nullptr and set int\_length to some default value.

* MyBigInt();

1. Write an overloaded constructor and perform deep copy (creation of dynamic memory).

* MyBigInt (int size);

HINT: you can write 1 constructor for task 1 and 2. (if you know it, you know it ;))

1. Write a copy constructor and perform deep copy. Print “Copy Constructor Called” and observer the scenarios where the copy constructor is called.

* MyBigInt (const MyBigInt & obj);

1. Write a member function to make a deep copy of the big\_int\_ of the passed MyBigInt obj into the big\_int\_ of the object which called this function.

* void assign(const MyBigInt & obj);.

1. Write a member function to display the big\_int\_ on screen. If big\_int\_ is empty, print “No Value Assigned”.

* void display();

1. Write a destructor to deallocate any dynamically allocated memory.

* ~ MyBigInt();

1. Write main() in the source.cpp to test all the functions of the MyBigInt class.

HINT: to call function of task 4, you need to follow something like this(not exactly):

//Create 2 objects of MyBigInt

obj2.assign(obj1);//call to function

**Note:**

* Deallocate all dynamically allocated memory.
* Do not use any string class built-in functions.
* Follow all the code indentation, naming conventions and code commenting guidelines.