**National University of Computer and Emerging Sciences**



**Lab Manual 06**

**Object Oriented Programming**

|  |  |
| --- | --- |
| Course Instructor | Syeda Tayyaba Bukhari |
| Lab Instructor (s) | Fariha Maqbool  Muhammad Usama |
| Section | BDS-2B |
| Semester | Spring 2022 |

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 a copy constructor
* Pass objects as parameters to functions

**TASK:**

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 compare the big\_int\_ of MyBigInt obj with the big\_int\_ of the object which called this function. Return 0 for equal, 1 for less than and 2 for greater than.

* int compareTo(const MyBigInt & obj);

HINT: first you need to check whether both objects are of same size or not. If yes only then you have to check the contents of arrays.

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.