**National University of Computer and Emerging Sciences**



**Lab Manual 04**

**Object Oriented Programming**

|  |  |
| --- | --- |
| Course Instructor | Ms. Arooj Khalil |
| Lab Instructor (s) | Ms. Fariha Maqbool  Ms. Amara Nasir |
| Section | BSE-2A |
| 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 a copy constructor
* Overloaded constructor
* Overloaded function
* Pass objects as parameters to functions
* Dynamic memory initialize in constructors
* Use \*this pointer

**TASK:**

Implement a class called **BiggerInt**. The BiggerInt 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.

You have to implement the following:

1. Write a default constructor and initialize big\_int\_ to nullptr.

* BiggerInt();

1. Write an overloaded constructor and perform deep copy.

* BiggerInt (const int \* obj, int size);

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

* BiggerInt (const BiggerInt & obj);

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

* void assign(const BiggerInt & obj);

1. Write a member function which will overload the above assign function and performs the same operations but the argument passed to this function is a pointer integer array.

* void assign(const int \* big\_int, int size);

1. Write a member function to append the big\_int\_ of the passed BiggerInt obj to the end of big\_int\_ of the object which called this function.

* void append(const BiggerInt & obj);

1. Write a member function which will overload the above append function and performs the same operations but the argument passed to this function is a pointer integer array.

* void append(const int\* big\_int, int size);

1. Write a member function to compare the big\_int\_ of BiggerInt 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 BiggerInt & obj);

1. Write a member function which overloads the above compareTo function and performs the same operations but the argument passed to this function is a pointer integer array.

* int compareTo(const int\* big\_int, int size);

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.

* ~ BiggerInt();

1. Write a suitable main() function in the driver.cpp to test all the functions of the BiggerInt class.

**Note:**

* Deallocate all dynamically allocated memory.
* Make separate my\_big\_int.h, my\_big\_int.cpp and driver.cpp files.
* Do not use any string class built-in functions except for strlen(), if required.
* Follow all the code indentation, naming conventions and code commenting guidelines.