National University of Computer and Emerging Sciences

Lab Manual 06
Object Oriented Programming

Course Instructor	Mr. Bismillah Jan
Lab Instructor (s)	Mr. Saif Ali Mr. Dilawar Shabbir
Section	BCS-2E
Semester	Spring 2021

Department of Computer Science
FAST-NU, Lahore, Pakistan

1.1 Objectives

After performing this lab, students shall be able to:

- Copy constructor
- Destructor
- this pointer
- Cascaded function calls.

TASK 1:

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();

2. Write an overloaded constructor and perform deep copy.
    - BiggerInt (const int * obj, int size);

3. 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);

4. 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);

5. 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);

6. 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);

7. 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);

8. 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);

9. 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);

10. Write a member function to display the big_int_ on screen. If big_int_ is empty, print "No Value Assigned".
    - void display();

11. Write a destructor to deallocate any dynamically allocated memory.
    - ~ BiggerInt();

12. 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.

TASK 2: (Cascading)

Implement a class **Time**. The Time class will have three data members:

- int hours;
- int minutes;
- int seconds;

You have to implement the following:

13. Write a default constructor.

14. Write an overloaded constructor.

15. Write all setters for hours, minutes, seconds such that each method of returns a reference to itself. (Cascading)

    Make sure that

    - Hours can never be greater than 24 and less than 0.
    - Minutes can never be greater than 59 and less than 0.
    - Seconds can never be greater than 59 and less than 0.

    Use **Conditional ? : Operator** to check validity.

    Whenever object is created your setter logic should be checked.

16. Write a member function getCurrentTime() that returns time.

17. Write all getters.

18. Write a suitable main() function to test all the functions of the Time class such that implementation of function cascading is clear.