**Lab#3**

**Objective:** To illustrate and hands-on Copy Constructor and Destructor.

**Constructor**: The copy constructor is a constructor which creates an object by initializing it with an object of the same class, which has been created previously.

The copy constructor is used to:

* Initialize one object from another of the same type.
* Copy an object to pass it as an argument to a function.
* Copy an object to return it from a function.

If a copy constructor is not defined in a class, the compiler itself defines one.If the class has pointer variables and has some dynamic memory allocations, then it is a must to have a copy constructor.

**Destructor**: A destructor is also a special function which is called when created object is deleted.Destructor" functions are the inverse of constructor functions. They are called when objects are destroyed (deallocated).

Several rules govern the declaration of destructors. Destructors:

* Do not accept arguments.
* Cannot specify any return type (including **void**).
* Cannot return a value using the **return** statement.
* Cannot be declared as **const**, **volatile**, or **static**. However, they can be invoked for the destruction of objects declared as **const**, **volatile**, or**static**.
* Can be declared as **virtual**.

**Lab Tasks:**

* Write codes, compile and run.
* Write output(s) of code in below given boxes.

**Program 1 (Code):**

**#include<iostream>**

**usingnamespace std;**

**classLine**

**{**

**public:**

**int get Length(void);**

**Line(int len);// simple constructor**

**Line (const Line&obj) ;// copy constructor**

**~Line();// destructor**

**private:**

**int\*ptr;**

**};**

**// Member functions definitions including constructor**

**Line::Line(int len)**

**{**

**cout<<"Normal constructor allocating ptr"<<endl;**

**// allocate memory for the pointer;**

**ptr=newint;**

**\*ptr=len;**

**}**

**Line::Line(const Line&obj)**

**{**

**cout<<"Copy constructor allocating ptr."<<endl;**

**ptr=newint;**

**\*ptr=\*obj.ptr;// copy the value**

**}**

**Line::~Line(void)**

**{**

**cout<<"Freeing memory!"<<endl;**

**deleteptr;**

**}**

**intLine::getLength(void)**

**{**

**return\*ptr;**

**}**

**void display(Line obj)**

**{**

**cout<<"Length of line : "<<obj.getLength()<<endl;**

**}**

**int main()**

**{**

**Line line(10); //We make object of class**

**display(line);**

**return0;**

**}**

**OUTPUT:**

