National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Computing

CS212: Object Oriented Programming

Class: BSCS-5AB

Lab 7: Dynamic Objects & Operator Overloading

Date: 04-04-2016

Time: 09:00 am – 12:00 pm / 2 – 5 pm

Instructor: Ms. Hirra Anwar

Lab 7: Dynamic Objects & Operator Overloading

Introduction

Operator overloading in C++ allows us to use some of the existing C++ operators with user defined classes. Dynamic allocation uses the concepts of heap. For dynamic memory allocation & de-allocation new and delete operators are used.

Objectives

To understand the concept of operator overloading by implementing few more operators.
To apply the concepts of dynamic memory allocation wrt class.

Tools/Software Requirement

Visual Studio

Description

Operator overloading is just simply another way to make a function call.

Example:

```
class CpClass
 {
    int *p;
    public:
    CpClass(int v=0)
    {
      p = new int;
      *p = v;
     }
    ~CpClass(){delete p;}
    CpClass& operator=(CpClass &);
 };
```

```
CpClass& CpClass::operator=(CpClass & obj)
 {
   *p = *obj.p;       // Left operand's private member initialized to that of right op.
   return *this;     // Return the left operand (the object)
 };
```

Non-member operators

In the previous lab examples, the operators may be members or non-members, and it doesn't seem to make much difference. This usually raises the question, "Which should I choose?" In general, if it doesn't make any difference, they should be members, to emphasize the association between the operator and its class. When the left-hand operand is always an object of the current class, this works fine.

However, sometimes you want the left-hand operand to be an object of some other class. A common place you'll see this is when the operators << and >> are overloaded for iostreams. In that case, the overloaded operator functions should be non-member. If the non-member function has to access the class data members, then you need to declare them as friend.

Lab Task 1

Create a class of your own choice using the heap concepts and implement constructor, destructor, member functions & non-member functions.

Lab Task 2

Overload the following operators in an Integer Class having data members left and right.

1. / operator
2. ^ operator

Lab Task 2

Overload the following operators using the Box class.
*= operator

1. != operator
2. >> operator
3. << operator
4. = (assignment operator)
5. Copy constructor

Make sure you implement dynamic memory allocation in your program.

You should implement them as member/non-member functions based on the requirement.

Deliverables

Source code of all the tasks with your names & reg. no along with comments in all programs should be shown to the lab engineer and upload on LMS.