**Heaven’s Light is Our Guide**



**Rajshahi University of Engineering and Technology**

**Department of Computer Science and Engineering**

**Course No:** CSE.1204

**Course Title:** Sessional based on CSE.1203 (Object Oriented Programming)

**Lab Report No:** 06

**Lab Report On:** Pointer & Static type variable and Operator Overloading.

**Submitted By** **Submitted To**

Md. Ariful Islam Md. Asifur Rahman

Roll No: 1803046 Lecturer

Section: A Dept. of CSE,RUET

Department: CSE

**Problem No:** 01

**Problem Statement:** Implementation of  **Pointer & Static type variable** and **Operator overloading** in the following classes.

**class** classtest **class** classtest1

|  |
| --- |
| int \*marks;  static int no\_of\_students;  static int total\_marks; |
| classtest1 ( );  classtest1 ( int);  void print\_marks ( );  classtest1 operator + ( classtest1& ) ;  ~classtest1 ( ); |

|  |
| --- |
| int \*marks; |
| classtest ( );  classtest ( int);  void print\_marks ( );  classtest operator + ( classtest& ) ;  ~classtest ( ); |

**Theory**

A **pointer type variable** is a variable that represents a location rather than a value of a data item. The syntax for dynamically allocate memory for a pointer variable **int \*m** is **m = new int( )**

and syntax for assign value to that allocated memory is

**\*m = value**

We must call a **destroyer** at the end of the program to avoid **memory leakage**.

A **static variable** is that variable that’s **lifetime** is the execution time of the program. That is never destroyed before finishing the program. In a class if we declare a static variable **must be initialized** outside of the class. The syntax of declaring a static variable is **static data\_type variable\_name = value**.

**Operator overloading** is the method by which we can change the function of some specific **operators** to do some different task. Operator overloading is just a type of function overloading with some additional rules. The syntax is

**Return\_type\_classname operator op ( argument list )**

**{**

**Function body**

**}**

In binary operators, the operator overloading function get called for the object that is immediately left to it.

**Source Code**

1. **main.cpp :**

|  |
| --- |
| #include <iostream>  #include "classtest.h"  #include "classtest1.h"  using namespace std;  int main ( )  {  cout<<"\n # Class 1 #\n"<<endl;  classtest s1(10);  classtest s2(18);  classtest s3;  s3=s1+s2;  s1.show();  s2.show();  s3.show();  cout<<"\n # Class 2 #\n"<<endl;  classtest1 st1(10);  st1.show1();  classtest1 st2(18);  st2.show1();  classtest1 st3(5);  st3.show1();  classtest1 st4;  st4=st1+st2;  st4.show1();  return 0;  } |

1. **.h file :**

**classtest.h**

|  |
| --- |
| #ifndef CLASSTEST\_H  #define CLASSTEST\_H  class classtest  {  private:  int \*marks;  public:  classtest();  classtest(int);  void show();  classtest operator+(classtest&);  ~classtest();  };  #endif // CLASSTEST\_H |

|  |
| --- |
| #ifndef CLASSTEST1\_H  #define CLASSTEST1\_H  class classtest1  {  private:  int \*marks;  static int no\_of\_student;  static int total\_marks;  public:  classtest1();  classtest1(int);  void show1();  classtest1 operator+(classtest1&);  ~classtest1();  };  #endif // CLASSTEST1\_H |

**classtest1.h**

1. **.cpp file**

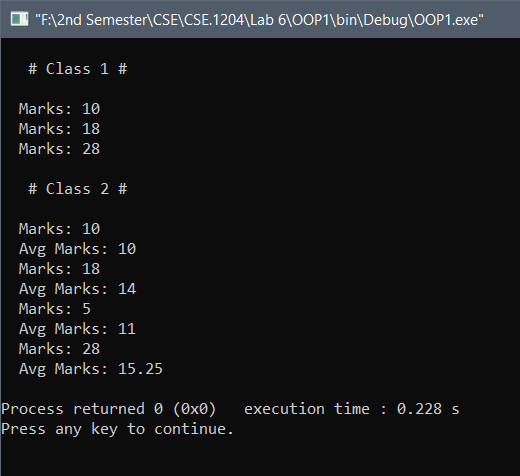
**classtest.cpp**

|  |
| --- |
| #include <iostream>  #include "classtest.h"  using namespace std;  classtest::classtest(){  marks=new int();  \*marks=0;  }  classtest::classtest(int a){  marks=new int();  \*marks=a;  }  void classtest::show(){  cout<<"Marks: "<<\*marks<<endl;  }  classtest classtest::operator+(classtest& ob){  int a;  a=\*marks+\*ob.marks;  classtest z(a);  return z;  }  classtest::~classtest(){  delete (marks);  } |

**classtest1.cpp**

|  |
| --- |
| #include<iostream>  #include "classtest1.h"  using namespace std;  int classtest1::no\_of\_student=0;  int classtest1::total\_marks=0;  classtest1::classtest1(){  marks=new int();  \*marks=0;  no\_of\_student+=1;  total\_marks+=0;  }  classtest1::classtest1(int a){  marks=new int();  \*marks=a;  no\_of\_student+=1;  total\_marks+=a;  }  void classtest1::show1(){  cout<<"Marks: "<<\*marks<<endl;  cout<<"Avg Marks: "<<1.0\*total\_marks/no\_of\_student<<endl;  }  classtest1 classtest1::operator+(classtest1& st){  int a;  a=\*marks+\*st.marks;  classtest1 Z(a);  no\_of\_student--;  return Z;  }  classtest1::~classtest1(){  delete (marks);  } |

**Output**

****

**Conclusion :** By our Course Teachers help and my knowledge about C and C++, I completed the program.

**# The End #**