ANJUMAN
COLLEGE OF ENGINEERING & TECHNOLOGY
(MANAGED BY : ANJUMAN HAMI-E-ISLAM, NAGPUR)

Object Oriented Programming Lab Manual

Computer Science
& Engineering
Department

Roll No:____
Name:_____
Sem:_____Section_____

CERTIFICATE

Certified that this file is submitted by

Shri/Ku._____

Roll No._____a student of _____ year of the course _____

_____ as a part of PRACTICAL/ORAL as prescribed by the Rashtrasant Tukadoji Maharaj Nagpur University for the subject_____ in the laboratory of _____during the academic year _____ and that I have instructed him/her for the said work, from time to time and I found him/her to be satisfactory progressive.

And that I have accessed the said work and I am satisfied that the same is up to that standard envisaged for the course.

Date:- Signature & Name of Subject Teacher Signature & Name of HOD

Anjuman College of Engineering and Technology

Vision

- To be a centre of excellence for developing quality technocrats with moral and social ethics, to face the global challenges for the sustainable development of society.

Mission

- To create conducive academic culture for learning and identifying career goals.
- To provide quality technical education, research opportunities and imbibe entrepreneurship skills contributing to the socio-economic growth of the Nation.
- To inculcate values and skills, that will empower our students towards development through technology.

Vision and Mission of the Department

Vision:

- To achieve excellent standards of quality education in the field of computer science and engineering, aiming towards development of ethically strong technical experts contributing to the profession in the global society.

Mission:

- To create outcome based education environment for learning and identifying career goals.
- Provide latest tools in a learning ambience to enhance innovations, problem solving skills, leadership qualities team spirit and ethical responsibilities.
- Inculcating awareness through innovative activities in the emerging areas of technology.

Program Educational Objectives (PEOs)

- The graduates will have a strong foundation in mathematical, scientific and engineering fundamentals necessary to formulate, solve and analyze engineering problem in their career.
- Graduates will be able to create and design computer support systems and impart knowledge and skills to analyze, design, test and implement various software applications.
- Graduates will work productively as computer science engineers towards betterment of society exhibiting ethical qualities.

Program Specific Outcomes (PSOs)

- Foundation of mathematical concepts: To use mathematical methodologies and techniques for computing and solving problem using suitable mathematical analysis, data structures, database and algorithms as per the requirement.
- Foundation of Computer System: The capability and ability to interpret and understand the fundamental concepts and methodology of computer systems and programming. Students can understand the functionality of hardware and software aspects of computer systems, networks and security.
- Foundations of Software development: The ability to grasp the software development lifecycle and methodologies of software system and project development.

PROGRAM: CSE	DEGREE: B.E
COURSE: Object Oriented Programming	SEMESTER: VI CREDITS: 2
COURSE CODE: BECSE302T	COURSE TYPE: REGULAR
COURSE AREA/DOMAIN: Object Oriented Programming	CONTACT HOURS: 2 hours/Week.
CORRESPONDING LAB COURSE CODE : BECSE302P	LAB COURSE NAME : Object Oriented Programming Lab

COURSE PRE-REQUISITES:

C.CODE	COURSE NAME	DESCRIPTION	SEM

LAB COURSE OBJECTIVES:

- Justify the philosophy of object-oriented design and the concepts of encapsulation, abstraction, inheritance, and polymorphism.
- Design, implement, test, and debug simple programs in an object-oriented programming language.
- Describe how the class mechanism supports Inheritance, Polymorphism.

COURSE OUTCOMES: Object Oriented Programming

After completion of this course the students will be able -

SNO	DESCRIPTION	BLOOM'S TAXONOMY LEVEL
CO.1	**Develop** program to illustrate basic concept of OOP features and C++ concept	LEVEL 3
CO.2	**Create** and implement program using unary and binary operator overloading	LEVEL 6
CO.3	**Write** program to implement concept of inheritance and polymorphism	LEVEL 6
CO.4	**Create** program to implement concept of abstract class and virtual functions	LEVEL 6
CO.5	**Develop** program using console I/O and file I/O	LEVEL 3
CO.6	**Develop** and implement program using exception handling and templates	LEVEL 3

Lab Instructions:

- Make entry in the Log Book as soon as you enter the Laboratory.
- All the students should sit according to their Roll Numbers.
- All the students are supposed to enter the terminal number in the Log Book.
- Do not change the terminal on which you are working.
- Strictly observe the instructions given by the Faculty / Lab. Instructor.
- Take permission before entering in the lab and keep your belongings in the racks.
- NO FOOD, DRINK, IN ANY FORM is allowed in the lab.
- TURN OFF CELL PHONES! If you need to use it, please keep it in bags.
- Avoid all horseplay in the laboratory. Do not misbehave in the computer laboratory. Work quietly.
- Save often and keep your files organized.
- Don't change settings and surf safely.
- Do not reboot, turn off, or move any workstation or PC.
- Do not load any software on any lab computer (without prior permission of Faculty and Technical Support Personnel). Only Lab Operators and Technical Support Personnel are authorized to carry out these tasks.
- Do not reconfigure the cabling/equipment without prior permission.
- Do not play games on systems.
- Turn off the machine once you are done using it.
- Violation of the above rules and etiquette guidelines will result in disciplinary action.

Continuous Assessment Practical

Exp No	NAME OF EXPERIMENT	Date	Sign	Remark
1	Write a C++ program using Static data member to record the occurrences of the entire object.			
2	Write a C++ program to use Multiple Constructor in a class for displaying complex value.			
3	Write a C++ program using Operator Overloading for overloading Unary minus operator.			
4	Write a C++ program using Operator overloading for overloading Binary operator.			
5	Write a C++ program to implement Multiple inheritance for multiplication of two numbers.			
6	Write a C++ program using Constructor in Derived classes to initialize alpha, beta and gamma and display corresponding values.			
7	Write a C++ program to display roll number, marks obtained in two subjects, sports weight, and total score of student using Virtual base class.			
8	Write a C++ program to display mean value of two numbers using Friend function.			
9	Write a C++ program to illustrate the use of a Vector Class Template for performing the scalar product of int type vector and float type vector.			
10	Write a C++ program for invoking function that generate Exception.			
11	Write a C++ program to display odd and even number using Command Line Argument.			
12	Implement a class string containing the following functions: - Overload + operator to carry out the concatenation of strings. - Overload = operator to carry out string copy. - Overload <= operator to carry out the comparison of strings. - Function to display the length of a string. - Function tolower( ) to convert upper case letters to lower case. - Function toupper( ) to convert lower case letters to upper case.			
13	Write a program to demonstrate the use of special functions, constructor and destructor in the Class template. The program is used to find the bigger of two entered numbers.			

CONTENTS

INTRODUCTION TO THE LAB

In this lab programming is done on Red Hat Linux using gcc compiler.

The C++ Programming Environment in Linux

The best way to learn a programming language is to try writing programs and test them on a computer. To do this, we need several pieces of software:

- An editor with which to write and modify the C++ program components or source code,
- A compiler with which to convert the source code into machine instructions which can be executed by the computer directly,
- A linking program with which to link the compiled program components with each other and with a selection of routines from existing libraries of computer code, in order to form the complete machine-executable object program,
- A debugger to help diagnose problems, either in compiling programs in the first place, or if the object program runs but gives unintended results.

There are several editors available for Linux (and Unix systems in general). Two of the most popular editors are emacs and vi. For the compiler and linker, we will be using the GNU g++ Compiler/linker, and for the debugger we will be using the GNU debugger gdb. For those that prefer an integrated development environment (IDE) that combines an editor, a compiler, a linking program and a debugger in a single programming environment (in a similar way to Microsoft Developer Studio under Windows NT), there are also IDEs available for Linux (e.g. V IDE, kdevelop etc.)

GNU Compiler Collection (GCC)

GCC stands for ―GNU Compiler Collection‖. GCC is an integrated distribution of compilers for several major programming languages. These languages currently include C, C++, Objective-C, Objective-C++, Java, Fortran , and Ada.

The abbreviation GCC has multiple meanings in common use. The current official meaning is ―GNU Compiler Collection‖, which refers generically to the complete suite of tools. The name historically stood for ―GNU C Compiler‖, and this usage is still common when the emphasis is on compiling C programs. Finally, the name is also used when speaking of the language independent component of GCC: code shared among the compilers for all supported languages.

The language-independent component of GCC includes the majority of the optimizers, as well as the ―back ends‖ that generate machine code for various processors.

The part of a compiler that is specific to a particular language is called the ―front end‖. In addition to the front ends that are integrated components of GCC, there are several other front ends that are maintained separately. These support languages such as Pascal, Mercury, and COBOL. To use these, they must be built together with GCC proper.

Most of the compilers for languages other than C have their own names. The C++ compiler is G++, the Ada compiler is GNAT, and so on. When we talk about compiling one of those languages, we might refer to that compiler by its own name, or as GCC. Either is correct.

Historically, compilers for many languages, including C++ and Fortran, have been implemented as ―preprocessors which emit another high level language such as C. None of the compilers included in GCC are implemented this way; they all generate machine code directly. This sort of preprocessor should not be confused with the C preprocessor, which is an integral feature of the C, C++, Objective-C and Objective-C++ languages.

GCC Command Options

When you invoke GCC, it normally does preprocessing, compilation, assembly and linking. The ―overall options‖ allow you to stop this process at an intermediate stage. For example, the –c option says not to run the linker. Then the output consists of object files output by the assembler.

Other options are passed on to one stage of processing. Some options control the preprocessor and others the compiler itself. Yet other options control the assembler and linker; most of these are not documented here, since you rarely need to use any of them.

The gcc program accepts options and file names as operands. Many options have multi-letter names; therefore multiple single-letter options may not be grouped: -dr is very different from `-d -r'.

You can mix options and other arguments. For the most part, the order you use doesn't matter. Order does matter when you use several options of the same kind; for example, if you specify –L more than once, the directories are searched in the order specified.

Many options have long names starting with `-f' or with `-W'—for example, -fmove-loopinvariants, -Wformat and so on. Most of these have both positive and negative forms; the negative form of –ffoo would be -fno-foo. This manual documents only one of these two forms, whichever one is not the default.

Compiling C++ Programs

C++ source files conventionally use one of the suffixes `.C', `.cc', `.cpp', `.CPP', `.c++', `.cp', or `.cxx'; C++ header files often use `.hh' or `.H'; and preprocessed C++ files use the suffix `.ii'. GCC recognizes files with these names and compiles them as C++ programs even if you call the compiler the same way as for compiling C programs (usually with the name gcc).

When you compile C++ programs, you may specify many of the same command-line options that you use for compiling programs in any language; or command-line options meaningful for C and related languages; or options that are meaningful only for C++ programs.

The Vi Editor

All Linux configuration files are written in plain English, easy to read and to adapt. You use a text-editor to write or make changes to such files. The two most popular, powerful, and unfortunately "difficult" text editors, both of which are found in every Linux are Vi and Emacs.

Most GUI-based editors, such as Kedit, are easier to manage. But don't make the mistake of thinking that a GUI-based editor is all you need. There are situations that crop up with Linux that require a text-mode editor -- in other words, when you don't have the luxury of accessing a GUI desktop at all. Vi and Emacs are the only tools that come with every Linux that work in text mode, so learning one or the other is mandatory.

Getting Started

To start Vi, open a terminal or console and simply type "vi" (without the quotation marks) followed by the name of any existing file or a new file you want to create.

Vi works in two main modes, one for editing text and the other for giving commands. To switch between the two modes you use the I (Insert) and Esc keys. The program opens in the Command mode, which is used for cursor movements, delete, cut, copy, paste, and saving changes.

The Insert mode is what you'll work in most of the time. You use it to make changes in an open file. Enter the Insert mode by pressing the I key. Newer Vi versions will display the word "INSERT" on the bottom line while you're in Insert mode.

Press the Esc key to switch Vi back to Command mode. As soon as you hit the Esc key the text "INSERT" on the bottom line disappears.
You save your changes to the open file from the Command mode. Press Shift-ZZ to save.

If you make a mistake when saving a file, such as pressing Ctrl-ZZ or closing Vi before saving the file, you'll end up with a swap file (akin to a DOS/Windows temp file) in addition to the original file. Usually the swap file will have the .swp extension. The original file will not contain the recent changes you made; attempting to reopen it will result in an error message.
The swap file is not readable but can be recovered by typing a command like this at the $ prompt and pressing Enter:

vi -r {your file name}

In some extreme cases, recovery is not possible. But in most cases, such as closing Vi before saving, a system crash, or a power failure, recovery works very well. After you recover, you must manually delete the swap file using a command like this at the $ prompt:

rm .{your file name}.swp

Common Vi Commands

Press Key(s):*	Function:
I	Insert text before the cursor
A	Insert text after the cursor
:	Switch to ex mode
$	Go to last place on the line
^	Go to first place on the line
W	Next word
B	Previous word
Shift-G	Last line of the file
20 Shift-G	Go to line 20
Y	Copy. (Note: Y3W = copy 3 words; Y3J = copy 4 lines.)
P	Paste
D	Cut
X	Delete character under the cursor

Steps for Creation of Program on Linux

1. Make a new file called "test" by opening a console and typing this line after the $ prompt and press Enter:

vi test

2. You'll get an empty console screen since Vi will start with the empty new file. Remember, Vi always starts Command mode, so press the **I (Insert)** key to enter the Insert mode. If you're ever not sure whether you're in Insert mode, you can always just hit **I** again.

3. Next, type the contents of your program.

4. Press the Esc key to return to the Command mode.

5. Save the file by pressing the **":wq"** key. This command save the file as well as exit from the file. But if you just want to save without exiting then press **":w"**.

6. Vi should close and you should see your $ prompt back in the console.

7. The program can be compiled by writing the following command on the $ prompt:

g++ filename.extension followed by pressing enter.

Eg: **g++ test.cpp**

8. If there are any errors in the program it will be displayed on the prompt with line numbers.

9. To remove the errors reopen the file you created. At the $ prompt, type this and press Enter: **vi test**

This time Vi opens up to the text in your file, not a blank screen.

10. Now save the file again by pressing Esc to enter the Command mode and recompile it.

11. If no errors then now run the program to get the output by typing this on $ prompt and press enter:

./a.out

12. Whenever you make any changes in your program, you need to save and recompile it before running it for the output.

13.Repeat the entire process again, for each program.

LAB REQUIREMENTS

Software Requirements:

For C++ Programming:

Linux Operating System

VI Editor or any other Text Editor

GCC Compiler

Hardware Requirements:

Computer nodes with proper configuration.

EXPERIMENT NO – 1

Aim: Write a C++ program using Static Data Member to record the occurrences of all the object.

Theory : We can define class members static using **static** keyword. When we declare a member of876a class as static it means no matter how many objects of the class are created, there is only one copy of the static member.

A static member is shared by all objects of the class. All static data is initialized to zero when the first object is created, if no other initialization is present. We can't put it in the class definition but it can be initialized outside the class as done in the following example by redeclaring the static variable, using the scope resolution operator **::** to identify which class it belongs to.

PROGRAM:

```
#include <iostream>

using namespace std;

class item

{

        static int count;

        int number;

        public:

                void getdata(int a)

                {

                        number=a;

                        count++;

                }

                void getcount(void)

                {

                        cout <<"Count: ";

                        cout <<count<<"\n";

                }

};

int item :: count;

int main()
```

```
{
        item a,b,c;
        a.getcount();
        b.getcount();
        c.getcount();
        a.getdata(100);
        b.getdata(200);
        c.getdata(300);
        cout<<" After Reading Data"<<"\n";
        a.getcount();
        b.getcount();
        c.getcount();
        return 0;
}
```

OUTPUT:

Conclusion: Thus we, have executed a C++ program using Static Data Member to record the occurrences of the entire object successfully.

Viva Voce Question

1. What is Object Oriented Programming?

2. What is the basic concept of OOP?

3. What is Inline Function?

4. Explain the concept of Reference variable.

5. Define Function Overloading.

6. Explain use of Scope Resolution operator.

7. Define Default argument used in C++.

_____
_____
_____
_____
_____

8. What are static members? When a data member of a class can be declared as static?

_____
_____
_____
_____
_____

9. What are the special characteristics of static members?

_____
_____
_____
_____
_____

10. What are the properties of static member functions?

_____
_____
_____
_____
_____

Signature of Subject Teacher

EXPERIMENT NO – 2

Aim: Write a C++ program to use Multiple Constructors in a Class for displaying Complex value.

Theory : A class **constructor** is a special member function of a class that is executed whenever we create new objects of that class.

A constructor will have exact same name as the class and it does not have any return type at all, not even void. Constructors can be very useful for setting initial values for certain member variables.

A class can have multiple constructors that assign the fields in different ways. Sometimes it's beneficial to specify every aspect of an object's data by assigning parameters to the fields, but other times it might be appropriate to define only one or a few.

PROGRAM:

```
#include <iostream>
using namespace std;
class complex
{
        float x,y;
public:
        complex(){}
        complex(float a)
        {
                x=y=a;
        }
        complex(float real, float imag)
        {
                x=real;
                y=imag;
        }
        friend complex sum(complex, complex);
        friend void show(complex);
};
complex sum(complex c1, complex c2)
{
        complex c3;
        c3.x=c1.x+c2.x;
```

```
        c3.y=c1.y+c2.y;
        return(c3);
}
void show(complex c)
{
        cout<<c.x<<"+j"<<c.y<<"\n";
}
int main()
{
        complex a(2.7, 3.5);
        complex b(1.6);
        complex c;
        c= sum(a,b);
        cout<<" A= ";
        show(a);
        cout<<"B=";
        show(b);
        cout<<"C=";
        show(c);
        return 0;
}
```

OUTPUT:

Conclusion: Thus we, have executed a C++ program to use Multiple Constructor in a Class for displaying Complex value successfully.

Viva Voce Question

1. What is Constructor?

_____
_____
_____
_____
_____

2. How a Constructor is invoked?

_____
_____
_____
_____
_____

3. What is the different type of Constructor?

_____
_____
_____
_____
_____

4. What are the special properties of Constructor Function?

_____
_____
_____
_____
_____

5. How dynamic initialization of objects achieved?

_____
_____
_____
_____
_____

6. Explain the function of Default Constructor?

7. What is Destructor?

8. Do we require a parameter for constructors?

9. What is a copy constructor?

10. Can we have more than one constructor in a class? If yes, explain the need for such a situation.

Signature of Subject Teacher

EXPERIMENT NO – 3

Aim: Write a C++ program using Operator Overloading for overloading Unary minus operator.

Theory: You can redefine or overload most of the built-in operators available in C++. Thus, a programmer can use operators with user-defined types as well.

Overloaded operators are functions with special names: the keyword "operator" followed by the symbol for the operator being defined. Like any other function, an overloaded operator has a return type and a parameter list.

```
Box operator+(const Box&);
```

declares the addition operator that can be used to **add** two Box objects and returns final Box object. Most overloaded operators may be defined as ordinary non-member functions or as class member functions. In case we define above function as non-member function of a class then we would have to pass two arguments for each operand as follows −

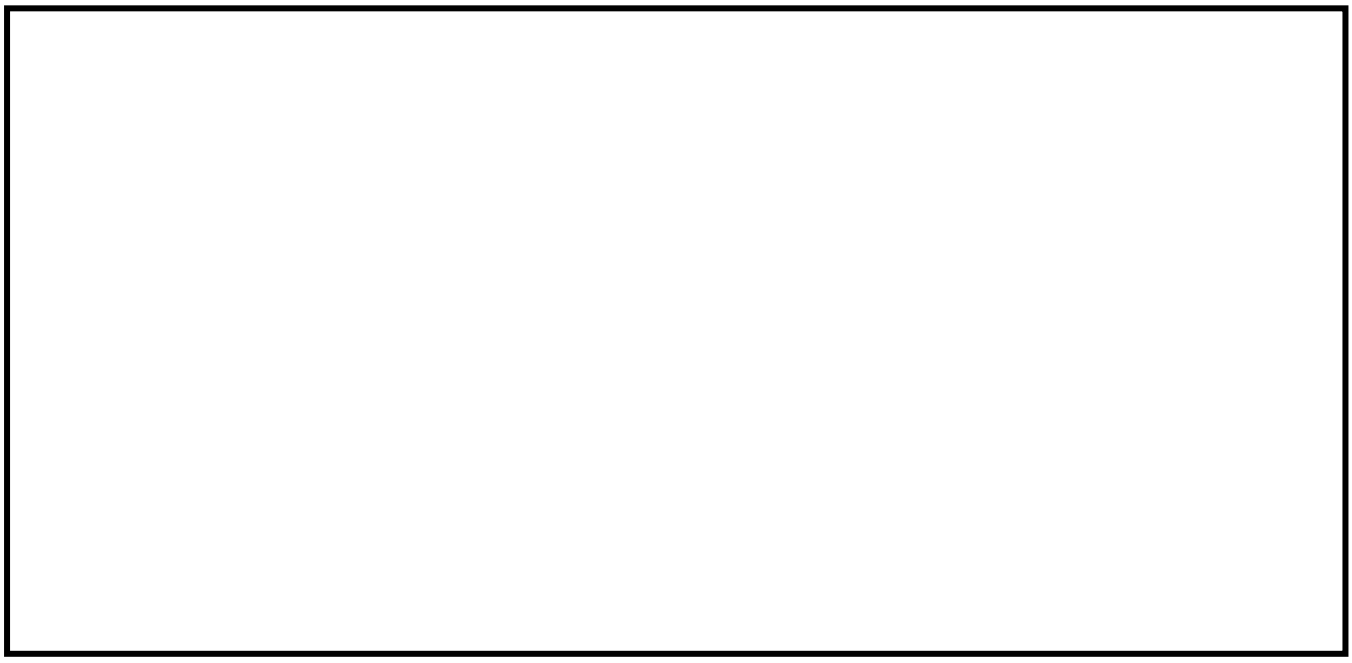
```
Box operator+(const Box&, const Box&);
```

PROGRAM:

```
#include<iostream>
using namespace std;
class space
{
        int x;
        int y;
        int z;
public:
        void getdata(int a, int b, int c);
        void display(void);
        void operator-();
};
void space :: getdata(int a, int b, int c)
{
        x=a;
        y=b;
        z=c;
}
void space :: display(void)
```

```
{
        cout<<"x= "<<x<<"";
        cout<<"y= "<<y<<"";
        cout<<"z= "<<z<<"\n";
}
void space :: operator-()
{
        x=-x;
        y=-y;
        z=-z;
}
int main()
{
        space s;
        s.getdata(10, -20, 30);
        cout <<"S : ";
        s.display();
        -s;
        cout <<"-S :";
        s.display();
        return 0;
}
```

OUTPUT:



Conclusion: Thus we, have executed a C++ program using Operator Overloading for overloading Unary minus operator successfully.

Viva Voce Question

1. What is Type Conversion?

   _____
   _____
   _____

2. What is operator overloading?

   _____
   _____
   _____

3. Why is it necessary to overload an operator?

   _____
   _____
   _____

4. Name the operators that cannot be overloaded in C++.

   _____
   _____
   _____

5. What is an operator function? Describe the syntax of an operator function.

   _____
   _____
   _____

6. How many arguments are required in the definition of an overloaded unary operator?

   _____
   _____
   _____

Signature of Subject Teacher

EXPERIMENT NO – 4

Aim: Write a C++ program using Operator overloading for overloading Binary operator.

Theory : The binary operators take two arguments and following are the examples of Binary operators. You use binary operators very frequently like addition (+) operator, subtraction (-) operator and division (/) operator.

Following example explains how addition (+) operator can be overloaded. Similar way, you can overload subtraction (-) and division (/) operators. In case of operator overloading of binary operators in C++ programming, the object on right hand side of operator is always assumed as argument by compiler.

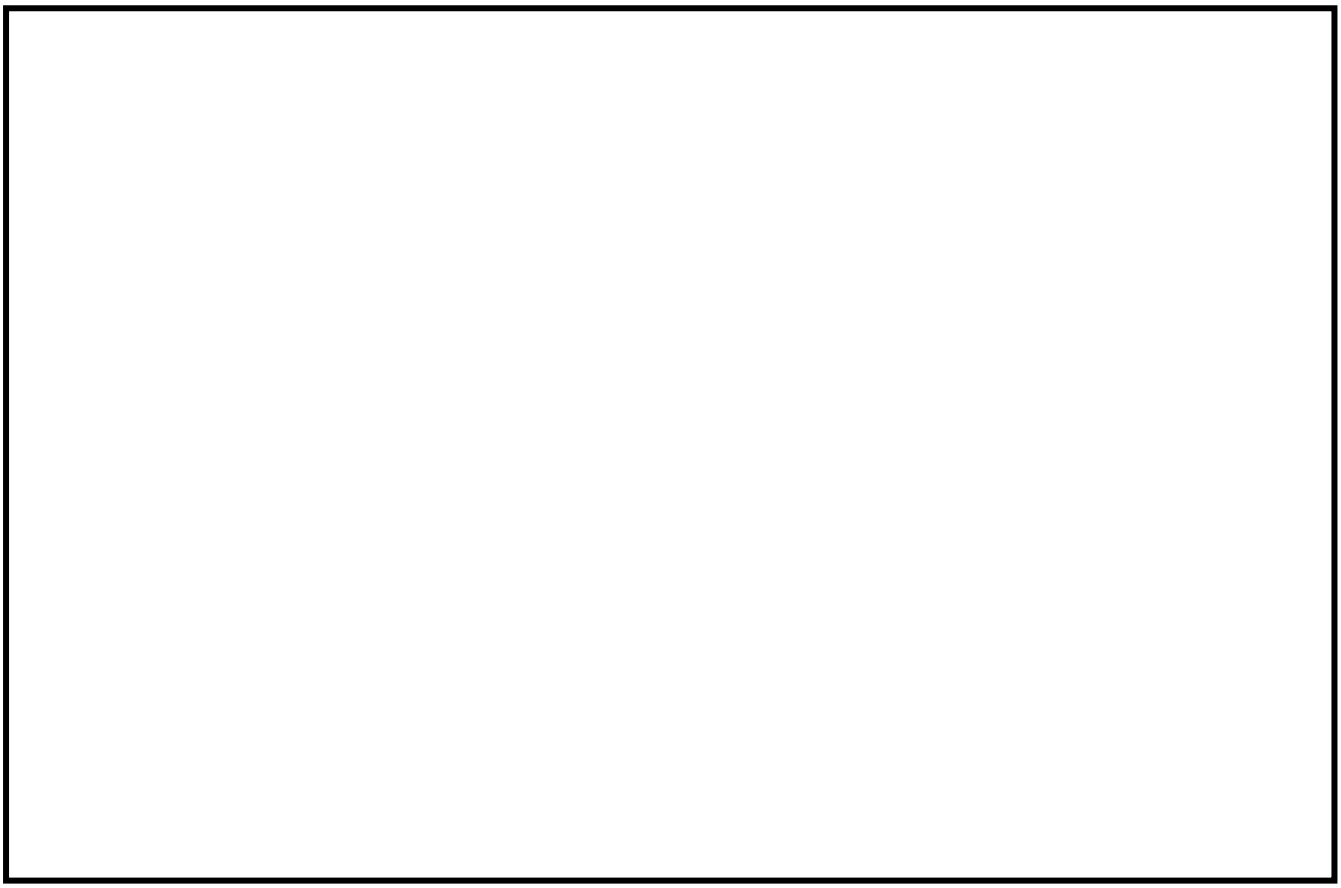
Then, this function returns the resultant complex number (object) to main() function which is displayed on to the screen.

PROGRAM:

```
#include <iostream>
using namespace std;
class Complex   {
float x, y;
public:
Complex()   {}
Complex(float real, float imag) {
x = real;
y = imag;
}
Complex operator +(Complex);
void display();
};
Complex Complex::operator +(Complex c)
{
Complex temp;
temp.x = x + c.x;
temp.y = y + c.y;
return (temp);
}
void Complex::display()
{
cout << x <<" + j"<< y <<"\n";
}
int main()
{
Complex c1, c2, c3;
c1 = Complex(2.5, 3.5);
```

```
c2 = Complex(1.6, 2.7);
c3 = c1 + c2;
cout <<"c1 = ";
c1.display();
cout <<"c2 = ";
c2.display();
cout <<"c3 = ";
c3.display();
return 0;
}
```

OUTPUT:



Conclusion: Thus we, have executed a C++ program using Operator overloading for overloading Binary operator successfully.

Viva Voce Question

1. Which keyword can be used for overloading?

   _____
   _____
   _____

2. What is overriding?

   _____
   _____
   _____

3. Which Keywords are used for overloading?

   _____
   _____
   _____

4. Which of the arithmetic operators can operate on string as well as numeric operands?

   _____
   _____
   _____

5. Is it possible to overload a constructor?

   _____
   _____
   _____

Signature of Subject Teacher

EXPERIMENT NO – 5

Aim: Write a C++ program to implement Multiple inheritance for multiplication of two numbers.

Theory : Multiple inheritance a feature of some object-oriented programming languages in which a class or an object inherits characteristics and properties from more than one parent class or object. This is contrary to the single inheritance property, which allows an object or class to inherit from one specific object or class. Although there are certain benefits associated with multiple inheritance, it does increase ambiguity and complexity when not designed or implemented properly.

PROGRAM:

```
#include<iostream>
using namespace std;
class M
{
        protected:
                int m;
        public:
                void get_m(int);
};
class N
{
        protected:
                int n;
        public:
                void get_n(int);
};
class P : public M, public N
{
        public:
                void display(void);
};

void M :: get_m(int x)
{
        m=x;
}
```

```
void N :: get_n(int y)
{
        n=y;
}
void P :: display()
{
        cout<<"m= "<<m<<"\n";
        cout<<"n= "<<n<<"\n";
        cout<<"m*n "<<m*n<<"\n";
}
int main()
{
        P p;
        p.get_m(10);
        p.get_n(20);
        p.display();
        return 0;
}
```

OUTPUT:

Conclusion: Thus we, have executed a C++ program to implement Multiple inheritance for multiplication of two numbers successfully.

Viva Voce Question

1. What does inheritance means in C++?

2. What are the advantages of using inheritance?

3. What are the different forms of inheritance? Give example of each.

4. What is default access modifier in a class?

5. Describe the syntax of multiple inheritance in C++. When do we use such a inheritance?

6. When to use the protected visibility specifier to a class member?

Signature of Subject Teacher

EXPERIMENT NO – 6

Aim: Write a C++ program using Constructor in Derived classes to initialize alpha, beta and gamma and display corresponding values.

Theory : A constructor plays a vital role in initializing an object. An important note, while using constructors during inheritance, is that, as long as a base class constructor does not take any arguments, the derived class need not have a constructor function. However, if a base class contains a constructor with one or more arguments, then it is mandatory for the derived class to have a constructor and pass the arguments to the base class constructor. Remember, while applying inheritance, we usually create objects using derived class. Thus, it makes sense for the derived class to pass arguments to the base class constructor. When both the derived and base class contains constructors, the base constructor is executed first and then the constructor in the derived class is executed.

In case of multiple inheritance, the base class is constructed in the same order in which they appear in the declaration of the derived class. Similarly, in a multilevel inheritance, the constructor will be executed in the order of inheritance.

The derived class takes the responsibility of supplying the initial values to its base class. The constructor of the derived class receives the entire list of required values as its argument and passes them on to the base constructor in the order in which they are declared in the derived class. A base class constructor is called and executed before executing the statements in the body of the derived class.

The header line of the derived-constructor function contains two parts separated by a colon (:). The first part provides the declaration of the arguments that are passed to the derived class constructor and the second part lists the function calls to the base class.

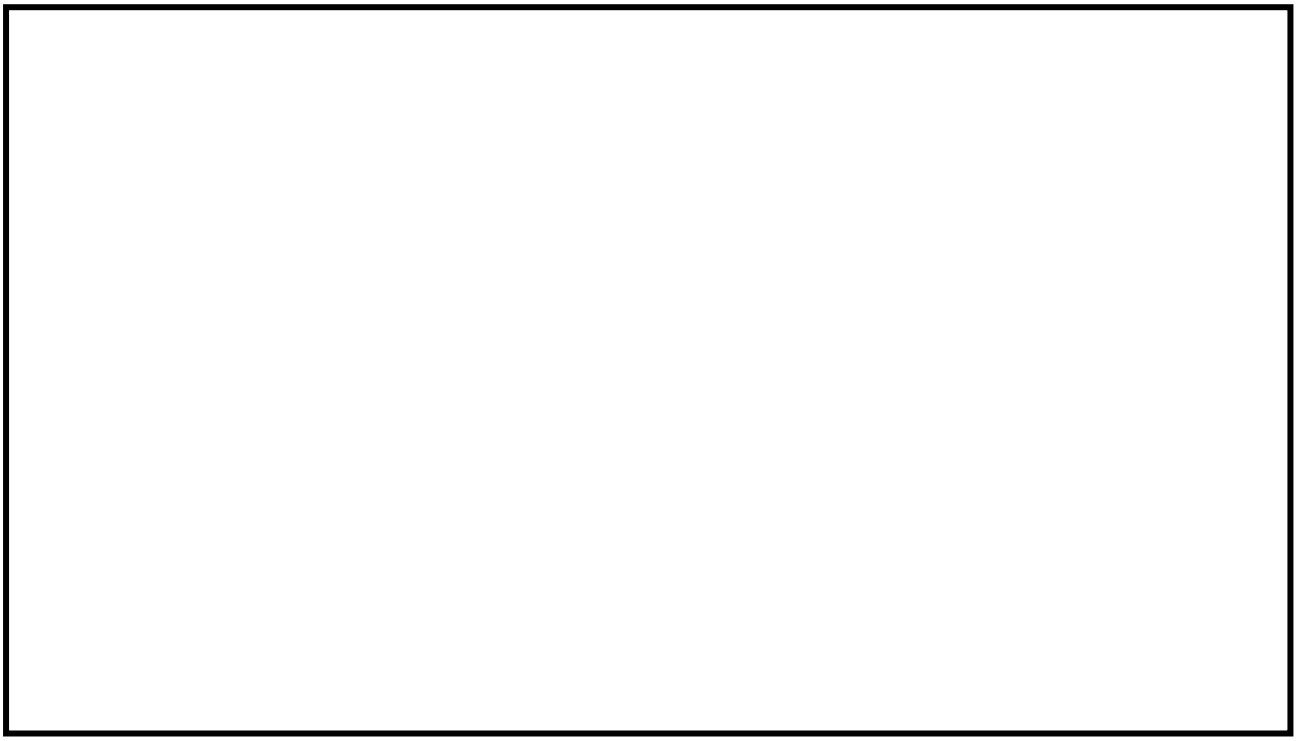
PROGRAM:

```
#include<iostream>
using namespace std;
class alpha
{
        int x;
        public:
                alpha(int i)
                {
                        x=i;
                        cout<<"Alpha Initialized";
                }
                void show_x()
                {
                        cout<<"\n x="<<x;
```

```
        }
};
class beta
{
    float y;
    public:
        beta(float j)
        {
            y=j;
            cout<<"\n Beta Initialized \n";
        }
        void show_y()
        {
            cout<<"\n y="<<y;
        }
};
class gamma:public beta,public alpha
{
    int m,n,c,d;
    public:
        gamma(int a,float b,int c,int d):alpha(a),beta(b)
        {
            m=c;
            n=d;
            cout<<"\n gamma initialized \n";
        }
        void show_mn()
        {
            cout<<"\n m="<<m;
            cout<<"\n n="<<n;
        }
};
int main()
{
    gamma g(5,7.65,30,100);
```

```
        g.show_x();
        g.show_y();
        g.show_mn();
        return 0;
}
```

OUTPUT:



Conclusion: Thus we, have executed a C++ program using Constructor in Derived classes to initialize alpha, beta and gamma and display corresponding values successfully.

Viva Voce Question

1. What are base class, sub class and super class?

2. Which specifier should be used for member function of a class?

3. How we declare a class as a virtual class?

4. In what order are the class constructors called when a derived class object is created?

5. What is virtual base class? When do we make a class virtual?

6. What is the general form of defining a derived constructor?

Signature of Subject Teacher

EXPERIMENT NO – 7

Aim: Write a C++ program to display roll number, marks obtained in two subjects, sports weight, and total score of student using Virtual base class.

Theory: An ambiguity can arise when several paths exist to a class from the same base class. This means that a child class could have duplicate sets of members inherited from a single base class. - C++ solves this issue by introducing a virtual base class. When a class is made virtual, necessary care is taken so that the duplication is avoided regardless of the number of paths that exist to the child class. When two or more objects are derived from a common base class, we can prevent multiple copies of the base class being present in an object derived from those objects by declaring the base class as virtual when it is being inherited. Such a base class is known as virtual base class. This can be achieved by preceding the base class' name with the word virtual.

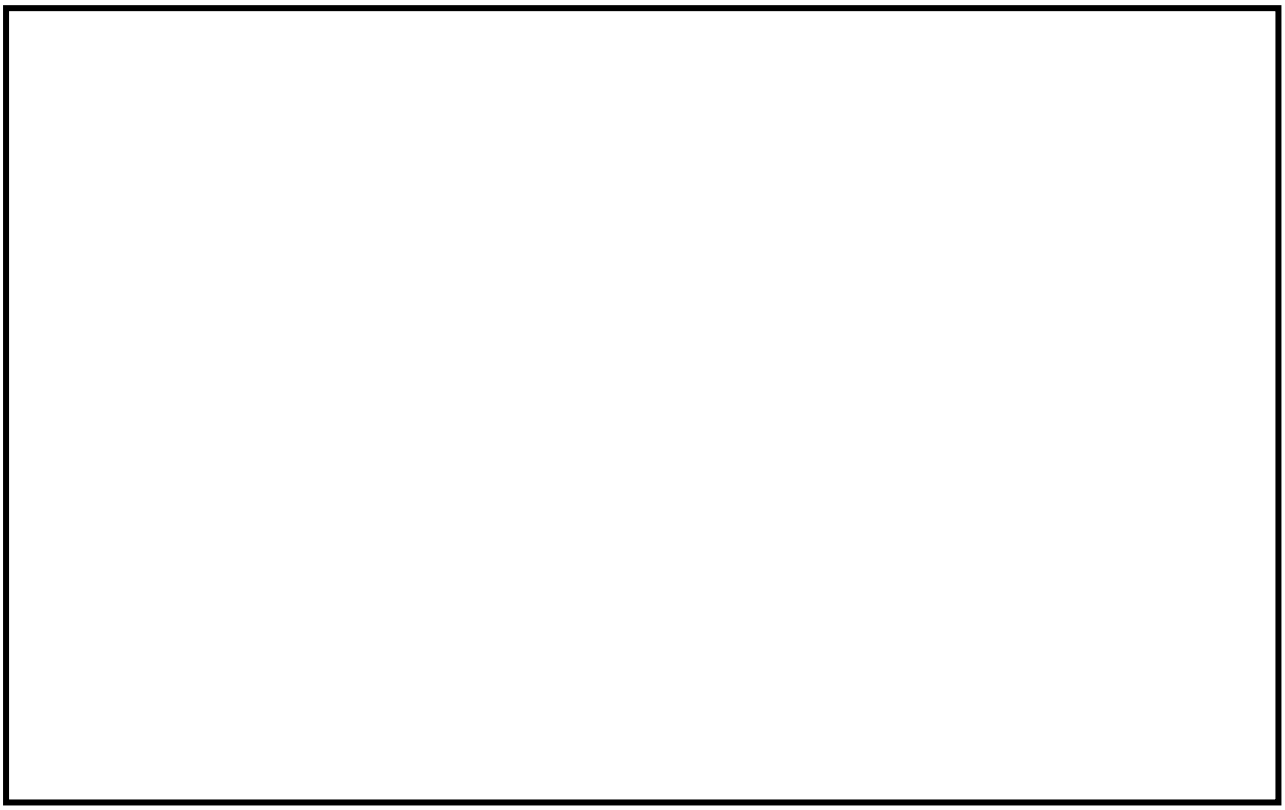
PROGRAM:

```
#include<iostream>

using namespace std;

class student

{

        protected:
                int rno;
        public:
                void get_no(int a)
                {
                        rno=a;
                }
                void put_no(void)
                {
                        cout<<"Roll No: "<<rno<<endl;
                }
};
class test : virtual public student
{
        protected:
                float part1,part2;
        public:
                void get_marks(float x,float y)
                {
                        part1=x;
                        part2=y;
                }
```

```
                void put_marks(void)
                {
                        cout<<"Marks obtained: "<<"\n"
                        <<"Part1= "<<part1<<"\n"
                        <<"Part2= "<<part2<<"\n";
                }
};


class sports: public virtual student
{
        protected:
                float score;
        public:
                void get_score(float s)
                {
                        score=s;
                }
                void put_score(void)
                {
                        cout<<"sports wt: "<<score<<"\n";
                }
};
class result: public test,public sports
{
        float total;
        public:
                void display(void);
};
void result :: display(void)
{
        total=part1+part2+score;
        put_no();
        put_marks();
        put_score();
        cout<<"total score: "<<total<<"\n";
}
int main()
{
        result std1;
        std1.get_no(1234);
        std1.get_marks(27.5,33.0);
        std1.get_score(6.0);
        std1.display();
        return 0;
```

```
}
```

OUTPUT:



Conclusion: Thus we, have executed a C++ program to display roll number, marks obtained in two subjects, sports weight, and total score of student using Virtual base class successfully.

Viva Voce Question

1. What is a Virtual Base Classes?

   _____
   _____
   _____

2. When do we make a class virtual?

   _____
   _____
   _____

3. How we declare a class as a virtual class?

   _____
   _____
   _____

4. What are the advantages of virtual base class?

   _____
   _____
   _____

5. What is an abstract class?

   _____
   _____
   _____

Signature of Subject Teacher

EXPERIMENT NO – 8

Aim: Write a C++ program to display mean value of two numbers using Friend function.

Theory : In object-oriented programming, a friend function, that is a "friend" of a given class, is a function that is given the same access as methods to private and protected data A friend function of a class is defined outside that class' scope but it has the right to access all private and protected members of the class. Even though the prototypes for friend functions appear in the class definition, friends are not member functions.

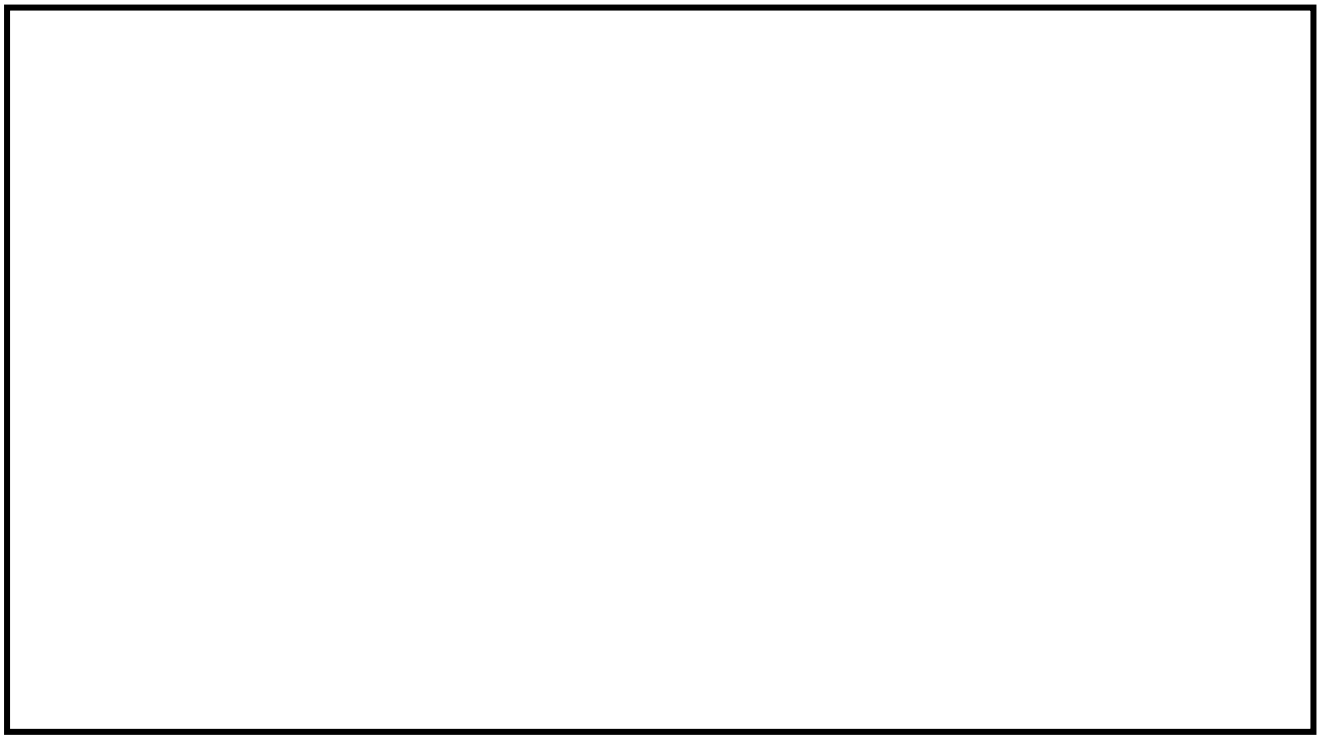
A friend can be a function, function template, or member function, or a class or class template, in which case the entire class and all of its members are friends.

To declare a function as a friend of a class, precede the function prototype in the class definition with keyword **friend**.

PROGRAM:

```
#include<iostream>
using namespace std;
class sample
{
        int a;
        int b;
        public:
                void setvalue()
                {
                        a=25;
                        b=40;
                }
                friend float mean(sample s);
};
float mean(sample s)
{
        return float(s.a+s.b)/2.0;
}
int main()
{
        sample X;
        X.setvalue();
        cout<<"Mean Value= "<<mean(X)<<"\n";
        return 0;
}
```

OUTPUT:



Conclusion: Thus we, have executed a C++ program to display mean value of two numbers using Friend function successfully.

Viva Voce Question

1. What is Friend function?

   _____
   _____
   _____

2. What are the merits and demerits of using friend function?

   _____
   _____
   _____

3. What are the special characteristics of friend function?

   _____
   _____
   _____

4. How to declare a function as friend function?

   _____
   _____
   _____

5. Can we make use of friend function in operator overloading?

   _____
   _____
   _____

Signature of Subject Teacher

EXPERIMENT NO – 9

Aim: Write a C++ program to illustrate the use of a Vector Class Template for performing the scalar product of int type vector and float type vector.

Theory : Templates are powerful features of C++ which allows you to write generic programs. In simple terms, you can create a single function or a class to work with different data types using templates. Templates are the foundation of generic programming, which involves writing code in a way that is independent of any particular type.

A template is a blueprint or formula for creating a generic class or a function. The library containers like iterators and algorithms are examples of generic programming and have been developed using template concept.

There is a single definition of each container, such as **vector**, but we can define many different kinds of vectors for example, **vector <int>** or **vector <string>**.

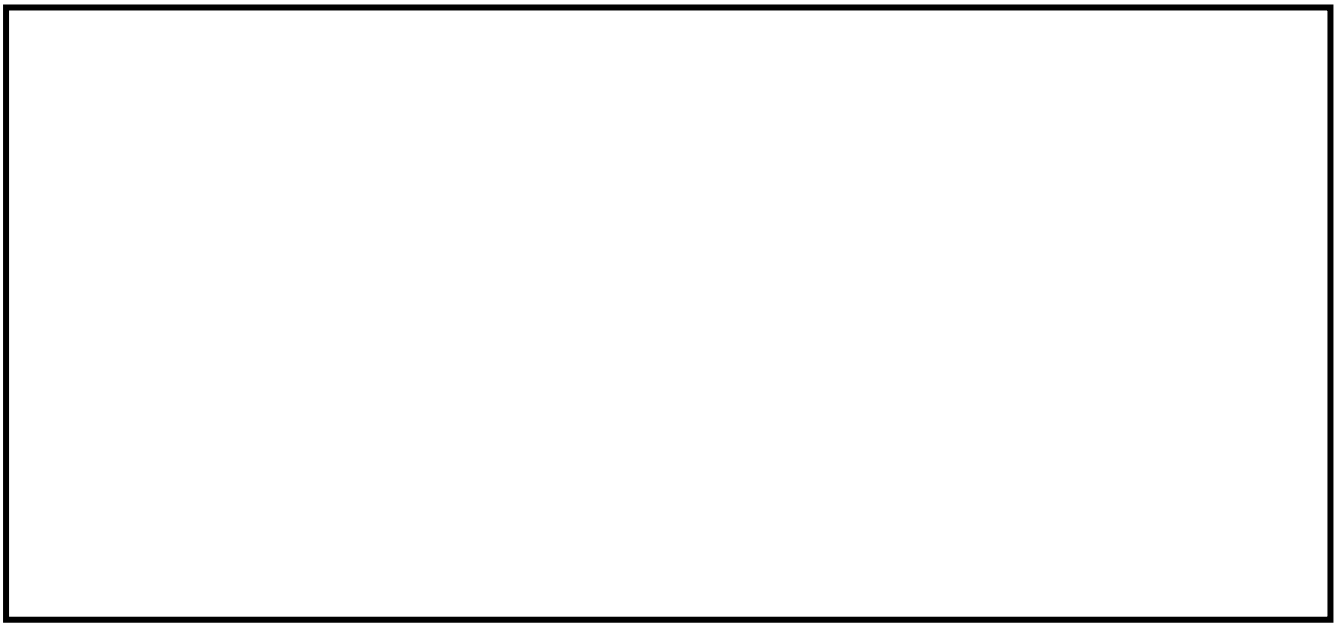
You can use templates to define functions as well as classes

PROGRAM:

```
#include<iostream>
using namespace std;
const int size=3;
template <class T>
class vector
{
        T *v;
        int size;

        public:
                vector(int)
                {
                        v=new T[size];
                        for(int i=0;i<size;i++)
                        {
                                v[i]=0;
                        }
                }
                vector(T *a)
                {
                        for(int i=0;i<size;i++)
                                v[i]=a[i];
                }
                T operator *(vector &y)
                {
```

```
                T sum=0;
                for(int i=0;i<size;i++)
                        sum+=this->v[i]*y.v[i];
                return sum;
        }
};
int main()
{
        int x[3]={1,2,3};
        int y[3]={4,5,6};


vector<int>v1(3);
        vector<int>v2(3);
        v1=x;
        v2=y;
        cout<<”v1=”;
        v1.display();
        cout<<”v2=”;
        v2.display();
        int R=v1*v2;
        cout<<"R= "<<R;
        return 0;
}
```

OUTPUT:



Conclusion: Thus we, have executed a C++ program to illustrate the use of a Vector Class Template for performing the scalar product of int type vector and float type vector successfully.

Viva Voce Question

1. What is generic programming? How is it implemented in C++?

   _____
   _____
   _____

2. Explain with the help of an example why templates are used in programming?

   _____
   _____
   _____

3. Distinguish between the term class template and template class.

   _____
   _____
   _____

4. Distinguish between overloaded functions and function templates?

   _____
   _____
   _____

5. A class (or function) template is known as parameterized class (or function). Comment.

   _____
   _____
   _____

Signature of Subject Teacher

EXPERIMENT NO – 10

Aim: Write a C++ program for invoking function that generate Exception.

Theory : An exception is a problem that arises during the execution of a program. A C++ exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero.

Exceptions provide a way to transfer control from one part of a program to another. C++ exception handling is built upon three keywords: **try, catch,** and**throw**.

- **throw** − A program throws an exception when a problem shows up. This is done using a **throw** keyword.
- **catch** − A program catches an exception with an exception handler at the place in a program where you want to handle the problem. The**catch** keyword indicates the catching of an exception.
- **try** − A **try** block identifies a block of code for which particular exceptions will be activated. It's followed by one or more catch blocks.+

Assuming a block will raise an exception, a method catches an exception using a combination of the **try** and **catch** keywords. A try/catch block is placed around the code that might generate an exception. Code within a try/catch block is referred to as protected code

You can list down multiple **catch** statements to catch different type of exceptions in case your **try** block raises more than one exception in different situations.

Throwing Exceptions

Exceptions can be thrown anywhere within a code block using **throw**statement. The operand of the throw statement determines a type for the exception and can be any expression and the type of the result of the expression determines the type of exception thrown.

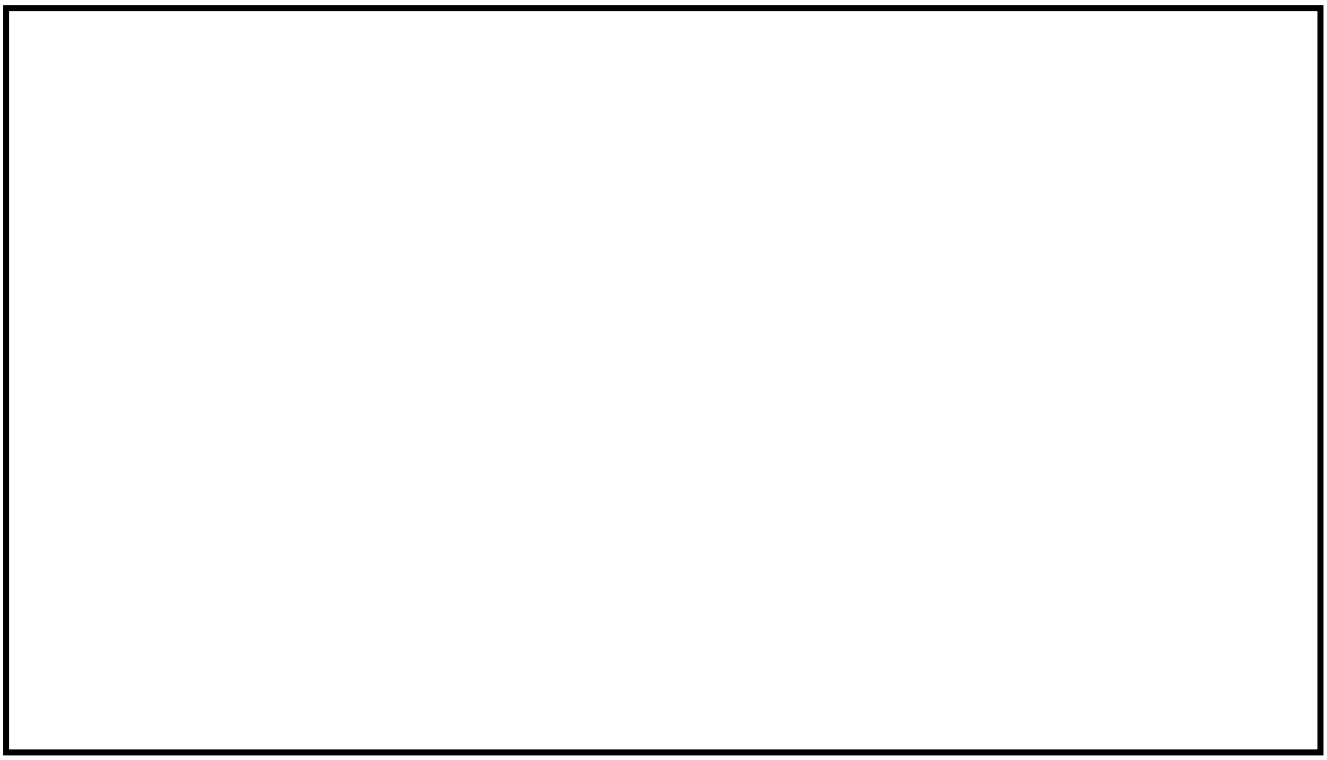
Catching Exceptions

The **catch** block following the **try** block catches any exception. You can specify what type of exception you want to catch and this is determined by the exception declaration that appears in parentheses following the keyword catch

PROGRAM:

```
#include<iostream>
using namespace std;
void divide(int x,int y,int z)
{
        cout<<"\n We are inside the function \n";
        if((x-y)!=0)
        {
                int R=z/(x-y);
                cout<<"Result= "<<R<<"\n";
        }
        else
        {
                throw(x-y);
        }
};
int main()
{
        try
        {
                cout<<"We are inside the try block \n";
                divide(10,20,30);
                divide(10,10,20);
        }
        catch(int i)
        {
                cout<<"Caught the exception \n";
        }
        return 0;
}
```

OUTPUT:



Conclusion: Thus we, have executed a C++ program for invoking function that generate Exception successfully.

Viva Voce Question

1. What is an Exception? How is an exception handled in C++?

   _____
   _____
   _____

2. What are the advantages of using exception handling mechanism in a program?

   _____
   _____
   _____

3. What should be placed inside a try block? Give syntax.

   _____
   _____
   _____

4. When should a program throw an exception?

   _____
   _____
   _____

5. When is a catch(…) handler used?

   _____
   _____
   _____

Signature of Subject Teacher

EXPERIMENT NO – 11

Aim: Write a C++ program to display odd and even number using Command Line Argument.

Theory : We know that every C++ program has atleast one function i.e **main( )**. Similar to other functions, **main( )** can also take arguments. These arguments are known as **command line arguments** and they are passed to the program at runtime. **main( )** actually takes two arguments : **int argc** is the argument count i.e the number of arguments passed to the program. **char *argv[ ]** is an array of strings where each string is a command line argument. The first string i.e **argv[0]** is the name of the program. Thus, there is atleast one argument always passed to main.

PROGRAM:

```
#include<iostream>
#include<fstream>
#include<stdlib.h>
using namespace std;
int main(int argc,char *argv[])
{
        int number[9]={11,22,33,44,55,66,77,88,99};
        if(argc!=3)
        {
                cout<<"argc="<<argc<<"\n";
                cout<<"error in arguments"<<"\n";
                exit(1);
        }
        ofstream fout1,fout2;
        fout1.open(argv[1]);
        if(fout1.fail())
        {
                cout<<"could not open the file"<<argv[1]<<"\n";
                exit(1);
        }
        fout2.open(argv[2]);
        if(fout2.fail())
        {
                cout<<"could not open the file"<<argv[2]<<"\n";
                exit(1);
```

```
        }
        for(int i=0;i<9;i++)
        {
                if (number[i]%2==0)
                        fout2<<number[i]<<""; //write to even file
                else
                        fout1<<number[i]<<""; //write to odd file
                }
        fout1.close();
        fout2.close();
        ifstream fin;
        char ch;
        for( int i=1;i<argc;i++)
        {
                fin.open(argv[i]);
                cout<<"contents of "<<argv[i]<<"\n";
                do
                {       fin.get(ch);
                        cout<<ch;       }
                while(fin);
                cout<<"\n\n";
                fin.close();
                }
        return 0;
}
```

OUTPUT:

```
acet@acet-desktop:~/code$ g++  commandl.cpp
acet@acet-desktop:~/code$ ./a.out commandl.cpp
Contents of ODD
11   33   55   77   99
Contents of EVEN
22  44  66  88
```

Conclusion: Thus we, have executed a C++ program to display odd and even number using Command Line Argument successfully.

Viva Voce Question

1. What is Command Line Argument?

   _____
   _____
   _____
   _____
   _____

2. How is the argument typed by the user get into the program?

   _____
   _____
   _____
   _____
   _____

3. What is file mode? Describe the various file mode options available.

   _____
   _____
   _____
   _____
   _____

Signature of Subject Teacher