```
        temp.hour=hour+t.hour;
        temp.minute=minute+t.minute;
        temp.second=second+t.second;
        return temp;
        }

};
void main()
{
time t1(20,30,40),t2(5,10,15),t3;
t3=t1.add(t2);
t3.show();
}
```

OUTPUT

```
25:40:55
```

MODULE - 5.2 : OPERATOR OVERLOADING

Program 1
Overloading ++ operator

```
#include <iostream.h>
class Increment
{
private:
      int data;
public:
      Increment()
            {
            data=0;
            }
      int display()
            {
            cout << data << endl;
            }
      void operator ++ ()
            {
            data ++;
            }

};
void main()
{
Increment obj1,obj2;
obj1.display();
obj2.display();
obj1++;
++obj1;
obj2++;
obj1.display();
obj2.display();
}
```

OUTPUT

```
0
0
2
1
```

```
// Program 2
// Overloading ++ operator with returning an object
#include <iostream.h>
class Increment
      {
      private:
            int data;
      public:
            Increment()
                  {
                  data=0;
                  }
            int display()
                  {
                  cout << data << endl;
                  }
            Increment operator ++ ()
                  {
                  Increment temp;
                  data ++;
                  temp.data = data;
                  return temp;
                  }
      };
void main()
      {
      Increment obj1,obj2;
      obj1.display();
      obj2.display();
      obj1++;
      obj2=obj1++;
      obj1.display();
      obj2.display();
      }
```

OUTPUT

```
0
0
2
2
```

```
// Program 3
// Overloading arithmetic + operator
#include <iostream.h>
class Data
      {
      private:
            int d;
      public:
            Data()
                  {d=0;}
            Data(int dt)
                  {d=dt;
```

```
        }
    void show()
        {
        cout << d << endl;
        }
    Data operator + (Data obj)
        {
        Data temp;
        temp.d = obj.d + d;
        return temp;
        }
};
d main()
{
Data d1(11),d2(22),d3;
d3=d1+d2;
d3.show();
}
```

OUTPUT

33

Program 4

Overloading arithmetic + operator

```
clude <iostream.h>
ss Height
  {
 private:
      int feet;
      float inches;
 public:
     Height()
         {
         feet=inches=0;
         }
     Height(int f, float i)
         {
         feet=f;
         inches=i;
         }
     void show()
         {
         cout << "Feet = " << feet << "\tInches = " << inches << endl;
         }
     Height operator + (Height h)
         {
         Height temp;
         temp.feet=feet + h.feet;
         temp.inches=inches+h.inches;
         if(temp.inches>=12)
             {
             temp.inches-=12;
```

```
                    temp.feet++;
                    }
                return temp;
                }
    };
void main()
    {
    Height H1(5,10),H2(6,11),H3;
    H3=H1+H2;
    H3.show();
    }
```

OUTPUT

Feet = 12 Inches = 9

```
// Program 5
// Overloading arithmetic + operator
// Evaluating H2 = H1 + 10.5 type of statement.
#include <iostream.h>
class Height
    {
    private:
        int feet;
        float inches;
    public:
        Height()
            {
            feet=inches=0;
            }
        Height(int f, float i)
            {
            feet=f;
            inches=i;
            }
        Height(float t)
            {
            feet = int(t);
            inches = 12*(t-feet);
            }
        void show()
            {
            cout << "Feet = " << feet << "\tInches = " << inches << endl;
            }
        Height operator + (Height h)
            {
            Height temp;
            temp.feet=feet + h.feet;
            temp.inches=inches+h.inches;
            if(temp.inches>=12)
                {
                temp.inches-=12;
                temp.feet++;
```

```
        }
    return temp;
    }

};
d main()
{
Height H1(5,10),H2;
H2=H1+10.5;
H2.show();
}
```

OUTPUT

Feet = 16 Inches = 4

Program 6

Overloading arithmetic + operator

Concatenating two string objects

```
clude <iostream.h>
clude <string.h>
ss String
{
private:
    char str[80];
public:
    String()
        {
        strcpy(str,"");
        }
    String(char s[80])
        {
        strcpy(str,s);
        }
    void display()
        {
        cout << str << endl;
        }
    String operator + (String s)
        {
        if(strlen(s.str)+strlen(str)<80)
            {
            String temp;
            strcpy(temp.str,str);
            strcat(temp.str,s.str);
            return temp;
            }
        else
            {
            cout << "Resultant String too large";
            }
        }
};
```

```
void main()
      {
      String s1("NICE Creates "),s2("Good Computer Programmers"),s3;
      s3=s1+s2;
      s3.display();
      }
```

OUTPUT

NICE Creates Good Computer Programmers

```
// Program 7
// Overloading comparision operator > and ==
#include <iostream.h>
class Data
      {
      private:
            int d;
      public:
            Data()
                  {
                  d=0;
                  }
            Data(int dt)
                  {
                  dt=0;
                  }
            int operator > (Data obj)
                  {
                  if (d > obj.d) return 1;
                  else return 0;
                  }
            int operator == (Data obj)
                  {
                  if (d == obj.d) return 1;
                  else return 0;
                  }
      };
void main()
      {
      Data t1(12),t2(24);
      if(t1==t2) cout << "Equal";
      else if(t1>t2) cout << "Greater";
      else cout << "Smaller";
      }
```

OUTPUT

Smaller

```
// Program 8
// Overloading += operator
#include <iostream.h>
class Height
      {
      private:
            int feet;
```

```
    float inches;
public:
    Height()
        {
        feet=inches=0;
        }
    Height(int f, float i)
        {
        feet=f;
        inches=i;
        }
    void show()
        {
        cout << "Feet = " << feet << "\tInches = " << inches << endl;
        }
    void operator += (Height h)
        {
        feet += h.feet;
        inches += h.inches;
        if(inches>=12)
            {
            inches-=12;
            feet++;
            }
        }
};
id main()
{
Height H1(5,10),H2(6,11);
H1+=H2;
H1.show();
}
```

OUTPUT

```
Feet = 12        Inches = 9
```

Program 9

Write a C++ program to overload following binary operators.
1.+ 2. - 3. * 4. /

```
clude <iostream.h>
ass Int
{
private:
    int num;
public:
    void set(int m)
        {num=m;}
    void show()
        {cout << num << endl;}
    Int operator + (Int t2)
        {Int temp;
        temp.num = num + t2.num;
```

```
            return temp;
            }
        Int operator - (Int t2)
            {Int temp;
            temp.num = num - t2.num;
            return temp;
            }
        Int operator * (Int t2)
            {Int temp;
            temp.num = num * t2.num;
            return temp;
            }
        Int operator / (Int t2)
            {Int temp;
            temp.num = num / t2.num;
            return temp;
            }
    };
void main()
    {
    Int i1,i2,i3;
    i1.set(12);
    i2.set(5);
    i3=i1+i2;
    i3.show();
    i3=i1-i2;
    i3.show();
    i3=i1*i2;
    i3.show();
    i3=i1/i2;
    i3.show();
    }
```

OUTPUT

```
17
7
60
2
```

```
/* Program 10.
   Write a C++ program to find area of two circles, which
have different radius. With the use of operator overloading
find which circle's area is greater.*/
#include <iostream.h>
#define PI 3.1415
class circle
    {
    private:
        int rad;
    public:
        circle(int r)
            {rad=r;}
        float area()
            {return PI*rad*rad;}
```

```
        int operator > (circle t)
              {
              if(rad > t.rad) return 1;
              else return 0;
              }
    };
void main()
    {
    circle c1(10),c2(20);
    cout << "Area of circle c1 : ";
    cout << c1.area() << endl;
    cout << "Area of circle c2 : ";
    cout << c2.area() << endl;
    if(c1>c2)
        {cout.<< "Area of circle c1 is greater than c2";}
    else
        {cout << "Area of circle c2 is greater than c1";}
    }
```

OUTPUT

```
Area of circle c1 : 314.149994
Area of circle c2 : 1256.599976
Area of circle c2 is greater than c1
```

/* Program 11

Declare a class to represent date (dd,mm,yy). Assume suitable member and data. Overload operator '-' and calculate age of a person. Use birth date and today's date. */

```
#include <iostream.h>
class date
    {
    private:
        int dd,mm,yy;
    public:
        date()
            {dd=mm=yy=0;}
        date(int d, int m, int y)
            {dd=d;
            mm=m;
            yy=y;
            }
        void read()
            {
            cout << "Enter date (dd mm yy) : ";
            cin >> dd >> mm >> yy;
            }
        void show()
            {cout << dd << ":" << mm << ":" << yy << endl;}
        date operator - (date t2)
            {int month[12]={31,28,31,30,31,30,31,31,30,31,30,31};
            int total_days=0;
            total_days+=((yy-1)-(t2.yy+1))*365;
            if(((t2.yy%4==0)&&(t2.yy%100!=0))||(t2.yy%400==0))
            month[1]=29;
```

```
                total_days+=month[t2.mm-1]-t2.dd;
                for(int i=t2.mm; i<12; ++i)
                      {
                      total_days+=month[i];
                      }
                if(((yy%4==0)&&(yy%100!=0))||(yy%400==0)) month[1]=29;
                else month[1]=28;
                for(i=0; i<mm-1; ++i)
                      {
                      total_days+=month[i];
                      }
                total_days+=dd;
                date temp;
                temp.yy=total_days/365;
                total_days-=(total_days/365)*365;
                for(i=0; i<12; ++i)
                      {
                      if(month[i]<total_days)
                            {
                            temp.mm++;
                            total_days-=month[i];
                            }
                      else break;
                      }
                temp.dd=total_days;
                return temp;
                }
      };
void main()
      {
      date birth(14,2,1976),today(23,3,2002),diff;
      diff=today-birth;
      diff.show();
      }
```

OUTPUT

```
7:125
(25 Year, 1 month and 7 Days)
```

```
/* Program 12
     Declare a class to represent complex (x,y). Overload +
operator to work on complex class' objects. */
#include <iostream.h>
class complex
     {
     private:
          int x,y;
     public:
          complex()
               {x=y=0;}
          complex(int xx, int yy)
               {x=xx;y=yy;
               }
```

```
        void read()
              {
              cout << "Enter x and y : ";
              cin >> x >> y;
              }
        complex operator + (complex c2)
              {
              complex temp;
              temp.x=x+c2.x;
              temp.y=y+c2.y;
              return temp;
              }
        void show()
              {
              cout << "x=" << x << " y=" << y << endl;
              }
    };
oid main()
    {
    complex p1(10,20),p2(30,40),p3;
    p3=p1+p2;
    p3.show();
    }
```

OUTPUT

X = 40 Y = 60

```
/ Program 13
/ Overload + operator to add two time objects
include <iostream.h>
lass time
    {
    private:
          int hour,minute,second;
    public:
          time()
                {hour=minute=second=0;}
          time(int h, int m, int s)
                {
                hour=h;
                minute=m;
                second=s;
                }
          void show()
                {
                cout << hour << ":" << minute << ":" << second << endl;
                }
          time operator + (time t)
                {
                time temp;
                temp.hour=hour+t.hour;
                temp.minute=minute+t.minute;
```

```
                temp.second=second+t.second;
                if(temp.second>=60)
                        {
                        temp.second-=60;
                        temp.minute++;
                        }
                if(temp.minute>=60)
                        {
                        temp.minute-=60;
                        temp.hour++;
                        }
                return temp;
                }
        };
void main()
        {
        time t1(20,30,40),t2(5,10,15),t3;
        t3=t1+t2;
        t3.show();
        }
```

OUTPUT

```
25:40:55
```

MODULE - 5.3 : INHERITANCE

```
// Program 1
// Simple Inheritance
#include <iostream.h>
class Base
      {
      protected:
            int x;
      public:
            void plus()
                  {
                  x++;
                  }
            void show()
                  {
                  cout << x << endl;
                  }
      };
class Derv : public Base
      {
      public:
            Derv()
                  {
                  x=0;
                  }
            void minus()
                  {
```

```
    }
};
oid main()
{
Derv obj;
obj.plus();
obj.plus();
obj.plus();
obj.minus();
obj.show();
}
```

OUTPUT

```
2
```

```
/ Program 2
/ Derived Class Constructor
include <iostream.h>
lass Base
{
protected:
      int x;
public:
      Base()
            {
            cout << "Base Class Constructor\n";
            x=0;
            }
      void plus()
            {
            x++;
            }
      void showBase()
            {
            cout << "x = " << x << endl;
            }
};
lass Derv : public Base
{
private:
      int y;
public:
      Derv() : Base()
            {
            cout << "Derived Class Constructor\n";
            y=0;
            }
      void minus()
            {
            x--;
            y--;
```

```
        }
    void showDerv()
        {
        cout << "y = " << y << endl;
        }
};
void main()
    {
    Derv obj;
    obj.plus();
    obj.plus();
    obj.plus();
    obj.minus();
    obj.showBase();
    obj.showDerv();
    }
```

OUTPUT

```
Base Class Constructor
Derived Class Constructor
x = 2
y = -1
```

```
// Program 3
/*
Create a class person having members name and age. Derive a
class student having member percentage. Derive another class
teacher having member salary. Write necessary member function
to initialize, read and write data. Also write the main
function.
*/
#include <iostream.h>
#include <string.h>
class person
    {
    private:
        char name[20];
        int age;
    public:
        person(char n[20], int a)
            {
            strcpy(name,n);
            age=a;
            }
        void display()
            {
            cout << "Name : " << name << endl;
            cout << "Age : " << age << endl;
            }
};
class student : public person
    {
    private:
        float perc;
    public:
```

```
    student(char n[20], int a, float p) : person(n,a)
        {
        perc=p;
        }
    void display()
        {
        cout << "Student" << endl;
        person :: display();
        cout << "Percentage : " << perc << endl;
        }

};
ss teacher : public person
{
private:
    float salary;
public:
    teacher(char n[20], int a, float s) : person(n,a)
        {
        salary=s;
        }
    void display()
        {
        cout << "Teacher" << endl;
        person :: display();
        cout << "Salary : " << salary << endl;
        }

};
id main()
{
student s("Dharmesh",18,78.9);
teacher t("D.H.Patel",45,10000);
s.display();
t.display();
}
```

OUTPUT

```
Student
Name : Dharmesh
Age : 18
Percentage : 78.900002
Teacher
Name : D.H.Patel
Age : 45
Salry : 10000
```

Program 4

rivate and public inheritance

This program will not compile and run. Watch the errors refully.

To understand this program, refer the diagram drawn in ur class-notebook. */

```
clude <iostream.h>
ss base
{
private: int a;
protected: int b;
public: int c;
};
```

```
class derv1 : public base
        {
        public:
             void display1()
                  {
                  cout << a << endl; //Error
                  cout << b << endl;
                  cout << c << endl;
                  }
        };
    class derv2 : private base
        {
        public:
             void display2()
                  {
                  cout << a << endl; //Error
                  cout << b << endl;
                  cout << c << endl;
                  }
        };
  void main()
        {
        derv1 d1;
        derv2 d2;

        cout << d1.a << endl; //Error
        cout << d1.b << endl; //Error
        cout << d1.c << endl;

        cout << d2.a << endl; //Error
        cout << d2.b << endl; //Error
        cout << d2.c << endl; //Error
        }
// Program 5
// Multiple Inheritance
#include <iostream.h>
class A
     {
     protected:
          int a;
     public:
          A(int x)
               {a=x;
               }
          void display()
               {cout << a << endl;
               }
   };
```

```
class B
    {
    protected:
          int b;
    public:
          B(int y)
                {
                b=y;
                }
          void display()
                {
                cout << b << endl;
                }
    };
class C : public A, public B
    {
    private:
          int c;
    public:
          C(int x, int y, int z) : A(x), B(y)
                {
                c=z;
                }
          void display()
                {
                A::display();
                B::display();
                cout << c << endl;
                }
    };
void main()
    {
    C obj(5,10,15);
    obj.display();
    }
```

OUTPUT

```
5
10
15
```

```
// Program 6
// Ambiguity in Multiple Inheritance
#include <iostream.h>
class A
    {
    public:
          void display()
                {
                cout << "Class A" << endl;
                }
    };
```

```
class B
      {
      public:
            void display()
                  {
                  cout << "Class B" << endl;
                  }
      };
class C : public A, public B
      {
      };
void main()
      {
      C obj;
      obj.display(); // Error
      obj.A::display();
      obj.B::display();
      }
```

OUTPUT

(If you remove second statment from main, then you will get following output)
Class A
Class B

```
/* Program 7
     Assume a class cricketer is declared. Declare a derived
class batsman from cricketer.
     Data member of batsman. Total runs, average runs and best
performance.
     Member functions. Input data, calculate average runs,
display data. */
#include <iostream.h>
class cricketer
     {
     private:
           char name[20];
           int total_matches;
     public:
           void read()
                 {
                 cout << "Name : "; cin >> name;
                 }
           void show()
                 {
                 cout << "Name : " << name << endl;
                 cout << "Total Matches : " << total_matches << endl;
                 }
     };
class batsman : public cricketer
     {
     private:
           int total_runs;
           int average_runs;
           int total_matches;
```

```
        int best_performance;
    public:
        void read()
            {
            cricketer :: read();
            cout << "Total Matches : "; cin >> total_matches;
            cout << "Total Runs : "; cin >> total_runs;
            cout << "Best Performance : "; cin >> best_performance;
            }
        void calculate()
            {average_runs = total_runs/total_matches;
            }
        void show()
            {
            cricketer :: show();
            cout << "Total Matches : " << total_matches << endl;
            cout << "Total Runs : " << total_runs << endl;
            cout << "Average : " << average_runs << endl;
            cout << "Best Performance : " << best_performance << endl;
            }
    };
void main()
    {
    batsman b;
    b.read();
    b.calculate();
    cout << "You have entered\n";
    b.show();
    }
```

OUTPUT

```
Name : Sachin
Total Matches : 208
Total Runs : 9876
Best Performance : 189
You have entered
Name : Sachin
Total Matches : 208
Total Runs : 9876
Average : 47
Best Performance : 189
```

// Program 8

// Example of Multiple and Multilevel Inheritance

```
#include <iostream.h>
#include <conio.h>
class college
    {
    private:
        char name[20];
        int year;
    public:
        void read()
            {
            cout << "Enter name : "; cin >> name;
            cout << "Year : "; cin >> year;
            }
        void show()
            {
            cout << "Name : " << name << endl;
            cout << "Year : " << year << endl;
```

```
                }
        };
    class teaching : public college
        {
        private:
            int no;
        public:
            void read()
                {
                college::read();
                cout << "Enter no of employees : "; cin >> no;
                }
            void show()
                {
                college::show();
                cout << "Number of employees : " << no << endl;
                }
        };
   class nonteaching : public college
        {
        private:
            int no;
            int holi;
        public:
            void read()
                {
                college::read();
                cout << "Enter no of employees : "; cin >> no;
                cout << "Enter holidays : " << holi;
                }
            void show()
                {
                college::show();
                cout << "Number of employees : " << no << endl;
                cout << "Number of holidays : " << holi << endl;
                }
     };
class ce : public teaching
     {
     private:
          char desg[20];
          char exp[20];
    public:
          void read()
               {
               teaching::read();
               cout << "Enter designation : "; cin >> desg;
               cout << "Enter experiance : "; cin >> exp;
               }
```

```
        void show()
              {
              teaching::show();
              cout << "Designation , " << desg << endl;
              cout << "Experiance : " << exp << endl;
              }

    };
lass cc : public teaching
    {
    private:
          char desg[20];
          char exp[20];
    public:
          void read()
              {
              teaching::read();
              cout << "Enter designation : "; cin >> desg;
              cout << "Enter experiance : "; cin >> exp;
              }
         void show()
              {
              teaching::show();
              cout << "Designation : " << desg << endl;
              cout << "Experiance : " << exp << endl;
              }

    };
lass it : public teaching
    {
    private:
          char desg[20];
    public:
          void read()
              {
              teaching::read();
              cout << "Enter designation : "; cin >> desg;
              }
          void show()
              {
              teaching::show();
              cout << "Designation : " << desg << endl;
              }

    };
oid main()
    {
    ec obj;
    obj.read();
    obj.show();
    }
```

```
// Program 9
// Program of hybrid inheritance
#include <iostream.h>
#include <conio.h>
class grandfather
      {
      private:
            int gr;
      public:
            grandfather()    {gr=0;}
            grandfather(long d)   {gr=d;}
            void show()
                  {
                  cout << "Grandfather : " << gr << endl;
                  }
      };
class uncle1 : public grandfather
      {
      private:
            long u1r;
      public:
            uncle1() {u1r=0;}
            uncle1(long d1, long d2) : grandfather(d1)
                  {u1r=d2;}
            void show()
                  {
                  cout << "Uncle 1 : " << u1r << endl;
                  }
      };
class uncle2 : public grandfather
      {
      private:
            long u2r;
      public:
            uncle2() {u2r=0;}
            uncle2(long d)
                  {u2r=d;}
            void show()
                  {
                  cout << "Uncle 2 : " << u2r << endl;
                  grandfather::show();
                  }
      };
class grandson : public uncle1, public uncle2
      {
      public:
            grandson(){}
            grandson(long l1, long l2, long l3) : uncle2(l2), uncle1(l3, l1)
                  {}
```

```
        void show()
             {
             uncle1 :: show();
             uncle2 :: show();
             }
    };
void main()
    {
    clrscr();
    grandson g(1000,2000,3000);
    g.show();
    getch();
    }
```

OUTPUT

```
Uncle 1 : 1000
Uncle 2 : 2000
Grandfather : 3000
```

MODULE - 5.4 : POINTER

```
// program 1
// pointer basics
#include <iostream.h>
void main()
    {
    int a=5,b=10;
    cout << a << " is stored at " << &a << endl;
    cout << b << " is stored at " << &b << endl;
    int *p;
    p=&a;
    cout << "p = " << p << endl;
    cout << "*p = " << *p << endl;
    p=&b;
    cout << "p = " << p << endl;
    cout << "*p = " << *p << endl;
    }
```

OUTPUT

```
5 is stored at 0x8fc8fff4
10 is stored at 0x8fc8fff2
p = 0x8fc8fff4
*p = 5
p = 0x8fc8fff2
*p = 10
```

```
// Program 2
// Arithmetic operator on pointers
#include <iostream.h>
void main()
    {
    int a=5,b=10,c,*p;
    cout << a << " is stored at " << &a << endl;
    cout << b << " is stored at " << &b << endl;

    p=&a;
    (*p)++;
    cout << "a = " << a << endl;

    p=&b;
    (*p)++;
    cout << "b = " << b << endl;
```

OUTPUT

```
5 is stored at 0x8fc8fff4
10 is stored at 0x8fc8fff2
a = 6
b = 11
*p = 17
```

```
        c=*p+a;
        p=&c;
        cout << "*p = " << *p << endl;
        }
```

```
// Program 3.1
// Pass with value
#include <iostream.h>
void main()
        {
        void swap(int,int);
        int a=5,b=10;
        cout << "Before Calling Function : a = " << a << " b = " << b << endl;
        swap(a,b);
        cout << "After Calling Function : a = " << a << " b = " << b << endl;
        }
void swap(int p1, int p2)
        {
        int temp;
        temp=p1;
        p1=p2;
        p2=temp;
        }
```

OUTPUT

```
Before Calling Function a = 5
b = 10
After Calling Function a = 5
b = 10
```

```
// Program 3.2
// Pass with Pointers
#include <iostream.h>
void main()
        {
        void swap(int*,int*);
        int a=5,b=10;
        cout << "Before Calling Function : a = " << a << " b = " << b << endl;
        swap(&a,&b);
        cout << "After Calling Function : a = " << a << " b = " << b << endl;
        }
void swap(int *p1, int *p2)
        {
        int temp;
        temp=*p1;
        *p1=*p2;
        *p2=temp;
        }
```

OUTPUT

```
Before Calling Function a = 5 b = 10
After Calling Function a = 10 b = 5
```

```
// Program 3.3
// Pass with reference
#include <iostream.h>
void main()
        {
```

```
    void swap(int&,int&);
    int a=5,b=10;
    cout << "Before Calling Function : a = " << a << " b = " << b << endl;
    swap(a,b);
    cout << "After Calling Function : a = " << a << " b = " << b << endl;
    }
void swap(int &p1, int &p2)
    {
    int temp;
    temp=p1;
    p1=p2;
    p2=temp;
    }
```

OUTPUT

```
Before Calling Function a = 5 b = 10
After Calling Function a = 10 b = 5
```

```
// Program 4
// Example of new operator
#include <iostream.h>
class Data
    {
    private:
          int d;
    public:
          set(int dt)
                {
                d=dt;
                }
          display()
                {
                cout << d << endl;
                }
    };
void main()
    {
    Data *p;
    p=new Data;
    p->set(12);
    p->display();
    }
```

OUTPUT

```
12
```

```
// Program 5
// An array of pointers to object
#include <iostream.h>
class Data
    {
    private:
          int d;
    public:
          void read()
```

```
        {
        cout << "Enter data : ";
        cin >> d;
        }
    void display()
        {
        cout << d << endl;
        }
};
void main()
{
Data *p[10];
int count=0,i;
char choice;
do
    {
    p[count]=new Data;
    p[count]->read();
    count++;
    cout << "More data ? (Y/N) ";
    cin >> choice;
    }
while(choice=='y' || choice=='Y');
for(i=0; i<count; ++i)
    {
    p[i]->display();
    }
}
```

OUTPUT

```
Enter data : 10
More data ? (Y/N) y
Enter data : 20
More data ? (Y/N) y
Enter data : 30
More data ? (Y/N) n
10
20
30
```

MODULE - 5.5 : virtual & friend Function/Class and this Pojnter

```
// Program 1
// Example of normal function
#include <iostream.h>
class Base
{
public:
    void show()
        {
        cout << "I am Base Class\n";
        }
};
class Derv1 : public Base
{
public:
    void show()
        {
        cout << "I am Derv1 Class\n";
        }
```

```
};
class Derv2 : public Base
{
public:
        void show()
                {
                cout << "I am Derv2 Class\n";
                }
};
void main()
{
Derv1 d1;
Derv2 d2;
Base *b;
b=&d1;
b->show();
b=&d2;
b->show();
}
```

OUTPUT

```
I am Derv1 Class
I am Derv2 Class
```

```
// Program 2
// Example of virtual function
#include <iostream.h>
class Base
{
public:
        virtual void show()
                {
                cout << "I am Base Class\n";
                }
};
class Derv1 : public Base
{
public:
        void show()
                {
                cout << "I am Derv1 Class\n";
                }

};
class Derv2 : public Base
{
public:
        void show()
                {
                cout << "I am Derv2 Class\n";
                }

};
void main()
{
```

```
        Derv1 d1;
        Derv2 d2;
        Base *b;
        b=&d1;
        b->show();
        b=&d2;
        b->show();
        }
```

OUTPUT

```
I am Derv1 Class
I am Derv2 Class
```

```
// Program 3
// Example of pure virtual function
#include <iostream.h>
class Base
      {
      public:
            virtual void show()=0;
      };
class Derv1 : public Base
      {
      public:
            void show()
                  {
                  cout << "I am Derv1 Class\n";
                  }
      };
class Derv2 : public Base
      {
      public:
            void show()
                  {
                  cout << "I am Derv2 Class\n";
                  }
      };
void main()
      {
      Derv1 d1;
      Derv2 d2;
      Base *b;
      b=&d1;
      b->show();
      b=&d2;
      b->show();
      }
```

OUTPUT

```
I am Derv1 Class
I am Derv2 Class
```

```
// Program 4
// Example of polymophism
#include <iostream.h>
class shape
      {
      public:
```

```
        virtual void read()=0;
        virtual void show()=0;
    };
class circle : public shape
    {
    private:
        float r;
    public:
        void read()
            {
            cout << "Enter radius : ";
            cin >> r;
            }
        void show()
            {
            cout << "Area of circle = " << 3.14 * r * r << endl;
            }
    };
class rectangle : public shape
    {
    private:
        float l,b;
    public:
        void read()
            {
            cout << "Enter Length of rectangle : ";
            cin >> l;
            cout << "Enter Breadth of rectagle : ";
            cin >> b;
            }
        void show()
            {
            cout << "Area of rectangle = " << l*b << endl;
            }
    };
class triangle : public shape
    {
    private:
        float b,h;
    public:
        void read()
            {
            cout << "Enter base of triangle : ";
            cin >> b;
            cout << "Enter height of triangle : ";
            cin >> h;
            }
        void show()
            {
```

```
        }
    void main()
    {
    shape *s[10];
    int count=0,i,choice,menu();
    choice=menu();
    while(choice!=4)
          {
          switch(choice)
          {
          case 1:
                s[count]=new circle;
                s[count]->read();
                count++;
                break;
          case 2:
                s[count]=new rectangle;
                s[count]->read();
                count++;
                break;
          case 3:
                s[count]=new triangle;
                s[count]->read();
                count++;
                break;
          default:
          cout << "Invalid Choice \n";
          }
          choice=menu();
          }
 for(i=0; i<count; ++i)
          {
          s[i]->show();
          }
}
int menu()
          {
          int ch;
          cout << "1 : Circle\n";
          cout << "2 : Rectangle\n";
          cout << "3 : Triangle\n";
          cout << "4 : Exit\n";
          cout << "Enter Your Choice : "
          cin >> ch;
          return ch;
          }
```

OUTPUT

```
1 : Circle
2 : Rectangle
3 : Triangle
4 : Exit
Enter Your Choice : 1
Enter radius : 10
1 : Circle
2 : Rectangle
3 : Triangle
4 : Exit
Enter Your Choice : 2
Enter Length of rectangle :
10
Enter Breadth of rectangle
: 20
1 : Circle
2 : Rectangle
3 : Triangle
4 : Exit
Enter Your Choice : 3
Enter base of triangle : 10
Enter height of triangle :
5
1 : Circle
2 : Rectangle
3 : Triangle
4 : Exit
Enter Your Choice : 4
Area of circle = 314
Area of rectangle = 200
Area of triangle = 25
```

```
// Program 5
// Ambiguity in multiple inheritance
// This program will not run, watch the errors carefully.
#include <iostream.h>
class Level1
      {
      public:
            int x;
      };
class Level21 : public Level1
      {
      };
class Level22 : public Level1
      {
      };
class Level3 : public Level21, public Level22
      {
      public:
            void set(int d)
                  {
                  x=d; // Error
                  }
            void show()
                  {
                  cout << x << endl; // Error
                  }
      };
void main()
      {
      Level3 obj;
      obj.set(12);
      obj.show()1
      }
/*    Program 5
      Avoid Ambiguity in multiple inheritance using virtual
base class */
#include <iostream.h>
class Level1
      {
      public:
            int x;
      };
class Level21 : virtual public Level1
      {
      };
class Level22 : virtual public Level1
      {
      };
```

```
class Level3 : public Level21, public Level22
    {
    public:
        void set(int d)
            {
            x=d;
            }
        void show()
            {
            cout << x << endl;
            }
    };
void main()
    {
    Level3 obj;
    obj.set(12);
    obj.show();
    }
```

OUTPUT

12

```
// Program 7
// friend function
#include <iostream.h>
class B;
class A
    {
    private:
        int x;
    public:
        A()
            {
            x=5;
            }
        friend void frifun(A,B);
    };
class B
    {
    private:
        int y;
    public:
        B()
            {
            y=5;
            }
        friend void frifun(A,B);
    };
void frifun(A obj1, B obj2)
    {
    cout << obj1.x + obj2.y << endl;
    }
```

```
void main()
      {
      A object1;
      B object2;
      frifun(object1,object2);
      }
// Program 8
// friend and operator overloading
// h2 = 10.5 + h1
#include <iostream.h>
class Height
      {
      private:
            int feet;
            float inches;
      public:
            Height()
                  {
                  feet=inches=0;
                  }
            Height(int f, float i)
                  {
                  feet=f;
                  inches=i;
                  }
            Height(float t)
                  {
                  feet=(int)t;
                  inches=12*(t-feet);
                  }
            void show()
                  {
                  cout << "Feet = " << feet << "\tInches = " << inches << endl;
                  }
            friend Height operator + (Height h1, Height h2)
                  {
                  Height temp;
                  temp.feet = h1.feet + h2.feet;
                  temp.inches = h1.inches + h2.inches;
                  if(temp.inches>=12)
                        {
                        temp.inches-=12;
                        temp.feet++;
                        }
                  return temp;
                  }
    };
```

OUTPUT

10

```
void main()
      {
      Height H1(5,10),H2;
      H2=10.5+H1;
      H2.show();
      }
```

OUTPUT

Feet=16 Inches=4

```
// Program 9
// friend as functional notation
// d2=square(d1);
#include <iostream.h>
class Data
      {
      private:
            int d;
      public:
            Data()
                  {
                  d=0;
                  }
            Data(int dt)
                  {
                  d=dt;
                  }
            void show()
                  {
                  cout << d << endl;
                  }
            friend Data square(Data obj)
                  {
                  Data temp;
                  temp.d = obj.d * obj.d;
                  return temp;
                  }
      };
void main()
      {
      Data d1(12),d2;
      d2=square(d1);
      d2.show();
      }
```

OUTPUT

144

```
// Program 10
// this pointer
#include <iostream.h>
class Data
     {
     private:
          int a;
```

```
        float b;
        char c;
    public:
        void show()
            {
            cout << this << endl;
            }
    };
void main()
    {
    Data t1,t2;
    t1.show();
    t2.show();
    }
```

OUTPUT

```
0x8fbcfffe
0x8fbcfff6
```

MODULE - 5.6 : Files & Streams

```
/* Program 1
 Simple File Program to write a line of text into a file on
disk.*/
#include <fstream.h>
void main()
    {
    ofstream obj("NICE.TXT");
    obj << "NICE makes good computer programmers\n";
    }
```

OUTPUT

After running this program, the contents of the file NICE.TXT should be :
NICE makes good computer programmers

```
// Program 2
// Simple File Program to read contents of a file on disk.
#include <fstream.h>
void main()
    {
    ifstream obj("NICE.TXT");
    char str[80];
    while(obj)
        {
        obj.getline(str,80);
        cout << str << endl;
        }
    }
```

OUTPUT

```
NICE makes good computer programmers
```

```
// Program 3
// Write into file sentence by sentence.
#include <fstream.h>
#include <string.h>
void main()
    {
    ofstream obj("WELCOME.TXT");
    char str[80]="Welcome to the world of programming at NICE";
```

OUTPUT

After running this program, the contents of the file WELCOME.TXT should be :
Welcome to the world of programming at NICE

```
    for(int i=0; i<strlen(str); ++i)
        {
        obj.put(str[i]);
        }
    }

// Program 4
// Reading from file sentence by sentence.
#include <fstream.h>
#include <string.h>
void main()
    {
    ifstream obj("WELCOME.TXT");
    char c;
    while(obj)
        {
        obj.get(c);
        cout << c;
        }
    }
```

OUTPUT

Welcome to the world of programming at NICE

```
// Program 5
// Reading/writing whole objects from/into file.
#include <fstream.h>
#include <string.h>
class student
    {
    private:
        int rollno;
        char name[20];
    public:
        student()
            {
            rollno=0;
            strcpy(name,"");
            }
        student(int r, char n[20])
            {
            rollno=r;
            strcpy(name,n);
            }
        void set(int r,char n[20])
            {
            rollno=r;
            strcpy(name,n);
            }
        void read()
            {
            cout << "Enter rollno : "; cin >> rollno;
```

```
    for(int i=0; i<strlen(str); ++i)
        {
        obj.put(str[i]);
        }
    }

// Program 4
// Reading from file sentence by sentence.
#include <fstream.h>
#include <string.h>
void main()
    {
    ifstream obj("WELCOME.TXT");
    char c;
    while(obj)
        {
        obj.get(c);
        cout << c;
        }
    }
```

OUTPUT

Welcome to the world of programming at NICE

```
// Program 5
// Reading/writing whole objects from/into file.
#include <fstream.h>
#include <string.h>
class student
    {
    private:
        int rollno;
        char name[20];
    public:
        student()
            {
            rollno=0;
            strcpy(name,"");
            }
        student(int r, char n[20])
            {
            rollno=r;
            strcpy(name,n);
            }
        void set(int r,char n[20])
            {
            rollno=r;
            strcpy(name,n);
            }
        void read()
            {
            cout << "Enter rollno : "; cin >> rollno;
```

```
        cout << "Enter name : "; cin >> name;
        }
    void show()
        {
        cout << "Rollno : " << rollno << endl;
        cout << "Name : " << name << endl;
        }
};
void main()
    {
    student s1(23,"Anang");
    student s2,s3,temp;
    s2.set(45,"Sachin");
    s3.read();
    ofstream obj1("NICESTUD.TXT");
    obj1.write((char*)&s1,sizeof(s1));
    obj1.write((char*)&s2,sizeof(s2));
    obj1.write((char*)&s3,sizeof(s3));
    obj1.close();
    cout << "Student Data\n";
    ifstream obj2("NICESTUD.TXT");
    obj2.read((char*)&temp,sizeof(temp));
    temp.show();
    obj2.read((char*)&temp,sizeof(temp));
    temp.show();
    obj2.read((char*)&temp,sizeof(temp));
    temp.show();
    }
```

OUTPUT

```
Enter rollno : 77
Enter name : Bhagirath
Student Data
Rollno : 23
Name : Anand
Rollno : 45
Name : Sachin
Rollno : 77
Name : Bhagirath
```

```
// Program 6
// Accessing random objects from file
#include <fstream.h>
#include <string.h>
class student
    {
    private:
        int rollno;
        char name[20];
    public:
        void show()
            {
            cout << "Rollno : " << rollno << endl;
            cout << "Name : " << name << endl;
            }
};
```

```
void main()
      {
      student s;
      int num,total_student,pos;
      ifstream fileobj("NICESTUD.TXT");
      fileobj.seekg(0,ios::end);
      total_student=fileobj.tellg()/sizeof(student);
      cout << "Total Students in File = " << total_student << endl;
      cout << "Enter the student number to view (1 to " << total_student << ") ";
      cin >> num;
      pos=(num-1)*sizeof(student);
      fileobj.seekg(pos);
      fileobj.read((char*)&s,sizeof(s));
      s.show();
      }
```

OUTPUT

```
Total Student in File = 3
Enter the student number to
view (1 to 3) 1
Rollno : 23
Name : Anand
```

```
// Program 7
// Command Line Argument
#include <iostream.h>
void main(int argc, char *argv[])
{
cout << "Total no. of arguments = " << argc << endl;
for(int i=0; i<argc; ++i)
{
cout << argv[i] << endl;
}
}
```

OUTPUT

```
Assume the file name is "0607.CPP"
C:\NICE\Material\CPP>0607 hello how are you
Total no. of arguments = 5
C:\NICE\Material\CPP\0607.EXE
hello
how
are
you
```

```
// Program 8
// Command Line Argument (mycopy source dest)
// Implementing DOS' copy command
#include <fstream.h>
void main(int argc, char *argv[])
      {
      char ch;
      if(argc!=3)
            {
            cout << "Invalid no. of arguments\n";
            cout << "Format : mycopy source dest\n";
            return;
            }
      ifstream ifs(argv[1]);
      ofstream ofs(argv[2]);
      while(ifs)
            {
            ifs.get(ch);
            ofs.put(ch);
            }
      }
```

OUTPUT

```
Assume the file name is
"MYCOPY.CPP"
Run the program as follows after
compilation, from DOS prompt.
C:\NICE\Material\CPP>mycopy
NICE.TXT DEST.TXT
```

```
/* program 9
     Store ten integer numbers in a file DATA. Read DATA file
and separate the odd and even numbers out of ten numbers and
then store all odd numbers in ODD file and even numbers in
EVEN file. Write a program which provides above facilities. */
#include <fstream.h>
void main()
{
int i,num;
ofstream ofs("DATA");
for(i=1; i<=10; ++i)
 {
 cout << "Enter number : ";
 cin >> num;
 ofs.write((char*)&num,sizeof(int));
 }
ofs.close();
ifstream ifs("DATA");
ofstream ofs1("EVEN");
ofstream ofs2("ODD");
for(i=1; i<=10; ++i)
 {
 ifs.read((char*)&num,sizeof(int));
 if(num%2==0) ofs1.write((char*)&num,sizeof(int));
 else ofs2.write((char*)&num,sizeof(int));
 }
ifs.close();
ofs1.close();
ofs2.close();
cout << "\nContents of DATA" << endl;
ifstream ifs1("DATA");
while(ifs1)
 {
 ifs1.read((char*)&num,sizeof(int));
 cout << num << "\t";
 }
ifs1.close();
cout << "\nContents of EVEN" << endl;
ifstream ifs2("EVEN");
while(ifs2)
 {
 ifs2.read((char*)&num,sizeof(int));
 cout << num << "\t";
 }
ifs2.close();
cout << "\nContents of ODD" << endl;
ifstream ifs3("ODD");
hile(ifs3)
 {
```

OUTPUT

```
Enter number : 1
Enter number : 2
Enter number : 3
Enter number : 4
Enter number : 5
Enter number : 6
Enter number : 7
Enter number : 8
Enter number : 9
Enter number : 0
Contents of DATA
1 2 3 4 5 6 7 8 9 0
Contents of EVEN
2 4 6 8 0
Contents of ODD
1 3 5 7 9
```

```
ifs3.read((char *)&num,sizeof(int));
cout << num << "\t";
 }
ifs3.close();
}
```

MODULE - 6 : Exception Handling & Template

```
// Program 1
// Exception
#include<iostream.h>
#include<conio.h>
void main()
{
int a;
cout<<"Enter the value of a(less than 100):";
try
      {
      cin>>a;
      if(a>100)
            throw a;
      }
      catch(int x)
      {
      cout<<"The value entered "<<x<<" is greater than 100";
      }
cout<<"The value entered "<<a<<" is less than 100 and is correct";
}
```

OUTPUT

```
Enter the value of a(less than 100) : 59
The value entered 59 is less than 100 and is correct
```

OUTPUT

```
Enter the value of a(less than 100) : 200
The value entered 200 is greater than 100
```

```
// Program 2
// Exception
#include<iostream.h>
#define N 10
class stack
{
private: int a[N],n;
public:
      stack()
            {n=0;}
      void push(int x)
            {if(n==10) throw 1;
            else a[n++];
            }
      int pop()
            {
            if(n<0) throw 2;
            else return(a[--n]);
            }
};
```