1. Takes two integer operands and one operator form the user, performs the operation and then prints the result.

```
#include<iostream>
using namespace std;
int a,b,ch;
int main()
{
	cout<<"Enter a:"<<endl;
	cin>>a;
	cout<<"Enter b:"<<endl;
	cin>>b;
	cout<<"enter\n1 to perform +\n2 to perform -\n3 to perform *\n4 to perform %\n5 to
perform /"<<endl;
	cin>>ch;
	switch(ch)
	{
		case 1:
			cout<<"Result for a+b is:"<<a+b<<endl;
			break;
		case 2:
			cout<<"Result for a-b is:"<<a-b<<endl;
			break;
		case 3:
			cout<<"Result for a*b is:"<<a*b<<endl;
			break;
		case 4:
			cout<<"Result for a%b is:"<<a%b<<endl;
			break;
		case 5:
		   cout<<"Result for a/b is:"<<a/b<<endl;
		   break;
		default:
			cout<<"please try again"<<endl;
	}
	return 0;
}
```

---

2. Generate all the prime numbers between 1 and n, where n is a value supplied by the user.

```
#include<iostream>
using namespace std;
int main()
{
   int n,i,j;
   cout<<"Enter the range of the N:";
   cin>>n;
   cout<<"\n"<<n<<": ";
   for(i=1;i<=n;i++)
   {
      for(j=2;j<=i;j++)
      {
         if(i%j==0)
         break;
      }
   if(i==j)
   cout<<j<<" ";
   }
return 0;
}
```

---

3. Searching an element in an array.

```
#include <iostream>
using namespace std;
int
main ()
{
  int a[50], i, n, pos, search;
  cout << "Enter how many elements to solve in an array" << endl;
  cin >> n;
  cout << "enter the elements into array" << endl;
  for (i = 0; i < n; i++)
```

```
   {
     cin >> a[i];
   }
  cout << "Enter search element" << endl;
  cin >> search;
  for (i = 0; i < n; i++)
   {
     if (a[i] == search)
          {
            pos = i;
            break;
          }
   }
  if (i == n)
   {
     cout << "Element not found in the list" << endl;
   }
  else
   {
     cout << "Element found at position: " << pos << endl;
   }
  return 1;
}
```

---

4. To find the factorial of a given integer.

```
#include <iostream>
using namespace std;
int n=1,fact=1,i;
int main()
{
   cout<<"enter element"<<endl;
   cin>>n;
   for(i=1;i<=n;i++)
   {
      fact=fact*i;
   }
   cout<<"factorial:"<<fact<<endl;
}
```

---

1. Write a program to demonstrate the Inline functions.

```
#include <iostream>
using namespace std;
inline int sum(int x, int y)
{
  return x + y;
}
int main() {
  int a, b, res;
  cout << "Enter the values: " << endl;
  cin >> a >> b;
  res = sum(a, b);
  cout << "Sum = " << res << endl;
  return 0;
}
```

---

2. Programs to understand different function call mechanism.
   a. call by reference

```
#include<iostream>
using namespace std;
void swap(int &x,int &y)
{
    int t;
    t=x;
    x=y;
    y=t;
}
```

```
int main()
{
   int a,b;
   cout<<"enter two values "<<endl;
   cin>>a>>b;
   cout<<"Before swapping"<<endl;
   cout<<"a="<<a<<" b="<<b<<endl;
   swap(a,b);
   cout<<"After swapping"<<endl;
   cout<<"a="<<a<<" b="<<b<<endl;
   return 1;
}
```

---

2. Programs to understand different function call mechanism.
   b. call by value

```
#include <iostream>
using namespace std;
void swap (int a, int b)
{
  int t;
  t=a;
  a=b;
  b=t;
  cout<<"After swapping:"<<endl;
  cout<<"a= "<<a<<"b= "<<b<<endl;
}
int main()
{
  int x,y;
  cout<<"Enter two values: "<<endl;
  cin>>x>>y;
  swap(x,y);
  return 1;
}
```

3. Programs to understand storage specifiers
a)Auto:-

```
#include<iostream>
using namespace std;
void autoStorageClass(){
cout<<"Demonstrating auto class\n";
auto a=32;
auto b=3.2;
auto c="Hello";
cout<<a<<"\n";
cout<<b<<endl;
cout<<c<<endl;
}
int main(){
autoStorageClass();
return 0;
}
```

3. Programs to understand storage specifiers
b)Extern:-

```
#include<iostream>
using namespace std;
int x;
void externStorageClass(){
cout<<"Demonstrating extern class\n";
extern int x;
auto b=3.2;
auto c="Hello";
cout<<"value of variable 'x' ";
cin>>x;
cout<<"Modified value of variable x"<<"declared as extern : \n"<<x;
}
int main(){
externStorageClass();
return 0;
```

```
}
```

---

3. Programs to understand storage specifiers
c)Register:-

```
#include<iostream>
#include<cstring>
using namespace std;
int main(){
char name[30];
register int i;
cout<<"Enter a string : ";
cin>>name;
cout<<"The revers string is : "<<endl;
for(i=strlen(name)-1;i>=0;i--){
cout<<name[i];
}
return 0;
}
```

---

3. Programs to understand storage specifiers
d) Static:-

```
#include<iostream>
#include<cstring>
using namespace std;
void printCount(void){
static int count=1;
cout<<"count = "<<count<<endl;
count=count+1;
}
int main(){
printCount();
printCount();
printCount();
printCount();
```

```
printCount();
return 0;
}
```

---

1. Write a Program to design a class having static member function Named showcount() which has the property of displaying the number of objects created of the class

```
#include<iostream>
using namespace std;
class test
{
private:
  int objno;
  static int objcnt;

public:
  test()
     {
       objno = ++objcnt;
     }
  ~test()
     {
       --objcnt;
     }
   void printobjnumber(void)
     {
       cout << "Object Number" << objno <<endl;
     }
static void printobjcount(void)
     {
       cout << "Object Count" << objcnt <<endl;
     }
};

int test::objcnt;
int main()
{
  test t1,t2;
  test::printobjcount();
  test t3;
  test::printobjcount();
```

```
  t1.printobjnumber();
  t2.printobjnumber();
  t3.printobjnumber();
  return 1;
}
```

---

2. Write a Program using class to process Shopping List for a Departmental Store. The list include details such as the Code No and Price of each item and perform the operations like Adding, Deleting Items to the list and Printing the Total value of a Order.

```
#include <iostream>
using namespace std;
const int m=50;
class ITEMS
{
    int itemcode[m];
    float itemPrice[m];
    int count;
    public:
        void CNT(void)
        {
            count=0;
        }
        void getitem(void);
        void displaysum(void);
        void remove(void);
        void displayItems(void);
};
void ITEMS::getitem(void)
{
    cout<<"Enter Item Code : ";
    cin>>itemcode[count];
    cout<<"Enter Item Cost : ";
    cin>>itemPrice[count];
    cout<<"\n";
    count++;
}
void ITEMS::displaysum(void)
{
    float sum=0;
```

```
    for(int i=0;i<count;i++)
    {
        sum=sum+itemPrice[i];
    }
    cout<<"\nTotal Value : "<<sum<<endl;
    cout<<"\n";
}
void ITEMS::remove(void)
{
    int a;
    cout<<"Enter Item Code : ";
    cin>>a;
    cout<<"\n";
    for(int i=0;i<count;i++)
    {
        if(itemcode[i]==a)
        {
            itemPrice[i]=0;
        }
    }
}
void ITEMS::displayItems(void)
{
    cout<<"\nCode\tPrice\n";
    for(int i=0;i<count;i++)
    {
        cout<<itemcode[i]<<"\t\t"<<itemPrice[i]<<"\n";
    }
    cout<<"\n";
}
int main()
{
    ITEMS order;
    order.CNT();
    int x;
    do
    {
        cout<<"You can do the following:\n";
        cout<<"Enter appropriate number\n";
        cout<<"\n1. Add an Item";
        cout<<"\n2. Display Total Value";
        cout<<"\n3. Delete an Item";
        cout<<"\n4. Display all Item";
        cout<<"\n5. Quit\n";
```

```
        cout<<"\nWhat is your Option : ";
        cin>>x;
        switch(x)
        {
            case 1:
                order.getitem();
                break;
            case 2:
                order.displaysum();
                break;
            case 3:
                order.remove();
                break;
            case 4:
                order.displayItems();
                break;
            case 5:
                break;
            default:
                cout<<"Error in input : Try again\n";
        }
    }while(x!=5);
    return 0;
}
```

---

1. Write a Program which creates & uses array of object of a class.(for eg. implementing the list of Managers of a Company having details such as Name, Age, etc..).

```
#include <iostream>
using namespace std;
class manager
{
    int Eid;
    char Name[30];
    int age;
    public:
    void getdata();
    void putdata();
};
void  manager::getdata()
{
```

```
    cout<<"Enter Employee Id: ";
    cin>>Eid;
    cout<<"Enter Employee Name: ";
    cin>>Name;
    cout<<"Enter Employee Age: ";
    cin>>age;
}

void manager::putdata()
{
    cout<<Eid<<" ";
    cout<<Name<<" ";
    cout<<age<<" ";
    cout<<endl;
}
int main()
{
    manager emp[30];
    int n,i;
    cout<<"Enter number of manager:";
    cin>>n;
    for(i=0;i<n;i++)
    {
        emp[i].getdata();
    }
        cout<<"Manager data " <<endl;
        for(i=0;i<n;i++)
        {
        emp[i].putdata();
        }
        return 1;

}
```

2. Write a Program to find Maximum out of Two Numbers using friend function. Note: Here one number is a member of one class and the other number is member of some other class

```
#include <iostream>
using namespace std;
class ABC;
class XYZ
{
   double x;
   public:
   void set_data(double a)
   {
      x=a;
   }
   friend void max(XYZ,ABC);
};
class ABC{
   double y;
   public:
   void set_data(double a)
   {
      y=a;
   }
   friend void max(XYZ,ABC);
};
void max(XYZ t1,ABC t2)
{
   if(t1.x>t2.y)
   cout<<t1.x;
   else
   cout<<t2.y;
}
int main()
{
   ABC _abc;
   XYZ _xyz;
   _xyz.set_data(29);
   _abc.set_data(37);
   max(_xyz,_abc);
   return 0;
```

}

---

1. Write a Program to swap private data members of classes Named as class_1, class_2 using friend function.

```
#include<iostream>
using namespace std;
class class_2;
class class_1
{
   int Value1;
   public:
   void indata(int a)
   {
      Value1=a;
   }
   void display()
   {
      cout<<Value1<<endl;
   }
friend void exchange(class_1 &,class_2 &);
};
class class_2
{
   int Value2;
   public:
   void indata(int a)
   {
      Value2=a;
   }
   void display()
   {
      cout<<Value2<<endl;
   }
friend void exchange(class_1 &,class_2 &);
};
void exchange(class_1 &x,class_2 &y)
{
   int t;
   t=x.Value1;
```

```
    x.Value1=y.Value2;
    y.Value2=t;
}
int main()
{
    class_1 c1;
    class_2 c2;
    c1.indata(100);
    c2.indata(200);
    cout<<"Values before exchange:"<<endl;
    c1.display();
    c2.display();
    exchange(c1,c2);
    cout<<"Values after exchange:"<<endl;
    c1.display();
    c2.display();
    return 0;
}
```

---

2. Write a Program to design a class complex to represent complex numbers. The complex class should use an external function (use it as a friend function) to add two complex numbers. The function should return an object of type complex representing the sum of two complex numbers.

```
#include<iostream>
using namespace std;

class complexNumber{
        float a;
        float b;
    friend class add;
    public:
        void getData(){
            cout<<"Enter number rational part = ";
            cin>>a;
            cout<<"Enter number irrational part = ";
            cin>>b;
        }
};

class add{
```

```
    int sumOfR;
    int sumOfI;
    public:
        void addi(complexNumber &c1,complexNumber &c2){
            sumOfR=c1.a+c2.a;
            sumOfI=c1.b+c2.b;
        }
        void print(complexNumber &c1,complexNumber &c2){
            cout<<"Complex number = "<<sumOfR<<"+"<<sumOfI<<"j"<<endl;
        }
};

int main(){
    complexNumber c1;
            complexNumber c2;
            add a;
            c1.getData();
            c2.getData();
            a.addi(c1,c2);
            a.print(c1,c2);

            return 0;
}
```

---

1. Write a Program using copy constructor to copy data of an object to another object.

```
#include <iostream>
using namespace std;
class Demo
{
    int num1;
    int num2;
    public:
        Demo(int n1,int n2)
        {
            num1=n1;
            num2=n2;
        }
        Demo(const Demo&n)
        {
            num1=n.num1;
```

```
        num2=n.num2;
    }
    void display()
    {
        cout<<"num1 = "<<num1<<endl;
        cout<<"num2 = "<<num2<<endl;
    }
};
int main()
{
   Demo obj1(10,20);
   Demo obj2(obj1);
   obj1.display();
   obj2.display();
   return 1;
}
```

---

2. Write a Program to allocate memory dynamically for an object of a given class using class's constructor.

```
#include <iostream>
using namespace std;

class someclass {
public:
  int data1;
  char data2;
  someclass() {
    cout << "constructer activated" << endl;
    cout<<"Enter data1: "<<endl;
    cin>>data1;
    cout<<"Enter data2: "<<endl;
    cin>>data2;
  }
  ~someclass() { cout << "Destructer activated" << endl; }
  void show() {
    cout << "data1= " << data1 << endl;
    cout << "data2= " << data2 << endl;
  }
};
```

```
int main() {
  someclass *ptr;
  ptr = new someclass;
  ptr->show();
  delete ptr;
  return 1;
}
```

```
//Write a Program to design a class to represent a matrix. The class should have the
//functionality to insert and retrieve the elements of the matrix

#include<iostream>
using namespace std;

class matrix
{
   int **p,d1,d2;
   public:
   matrix(int x,int y);
   void get_element(int i,int j,int value)
   {
      p[i][j]=value;
   }
   int & put_element(int i,int j)
   {
      return p[i][j];
   }

};

matrix :: matrix(int x,int y)
{
   d1=x;
   d2=y;
   p=new int*[d1];
   for(int i=0;i<d1;i++)
   p[i]=new int[d2];
}

int main()
{
```

```
    int m,n;
    cout<<"Enter the size of the matrix:"<<endl;
    cin>>m>>n;
    matrix A(m,n);
    cout<<"Enter matrix elements rowwise "<<endl;
    int i,j,value;
    for(i=0;i<m;i++)
    {
        for(j=0;j<n;j++)
        {
            cin>>value;
            A.get_element(i,j,value);
        }
    }
    cout<<A.put_element(0,1);
    return 0;
}
```

// Write a program to design a class representing complex numbers and having the functionality
// of performing addition & multiplication of two complex numbers using operator overloading.

```
#include<iostream>
using namespace std;

class complex
{
    private:
    float real;
    float imag;
    public:
    complex()
    {
        real=imag=0.0;
    }
    void getdata()
    {
        cout<<"enter real value:"<<endl;
        cin>>real;
        cout<<"enter imag value:"<<endl;
        cin>>imag;
    }
```

```
    void outdata(string msg)
    {
        cout<<endl<<msg<<"("<<real;
        cout<<","<<imag<<")"<<endl;
    }

    friend complex operator*(complex c1,complex c2);
    friend complex operator+(complex c1,complex c2);
};

complex operator *(complex c1,complex c2)
{
    complex temp;
    temp.real=c1.real*c2.real-c1.imag*c2.imag;
    temp.imag=c1.real*c2.imag+c1.imag*c2.real;
    return temp;
}

complex operator +(complex c1,complex c2)
{
    complex temp;
    temp.real=c1.real+c2.real;
    temp.imag=c1.imag+c2.imag;
    return temp;
}
int main()
{
    complex c1,c2,c3,c4;
    cout<<"enter 1st complex number:"<<endl;
    c1.getdata();
    cout<<"enter 2nd complex number:"<<endl;
    c2.getdata();
    c3=c1*c2;
    c3.outdata("c3=c1*c2");
    c4=c1+c2;
    c4.outdata("c4=c1+c2");
    return 0;
}
```

```
// Write a Program to overload operators like *, <<, >> using friend function. The following
// overloaded operators should work for a class vector.
#include <iostream>
using namespace std;

const int size = 3;

class vector
{
    int v[size];

public:
    vector();
    vector(int *x);
    friend vector operator*(int a, vector b);
    friend vector operator*(vector b, int a);
    friend istream &operator>>(istream &, vector &);
    friend ostream &operator<<(ostream &, vector &);
};

vector ::vector()
{
    for (int i = 0; i < size; i++)
        v[i] = 0;
}

vector ::vector(int *x)
{
    for (int i = 0; i < size; i++)
        v[i] = x[i];
}

vector operator*(int a, vector b)
{
    vector c;
    for (int i = 0; i < size; i++)
        c.v[i] = a * b.v[i];
    return c;
}

vector operator*(vector b, int a)
```

```
{
    vector c;
    for (int i = 0; i < size; i++)
        c.v[i] = b.v[i] * a;
    return c;
}

istream &operator>>(istream &din, vector &b)
{
    for (int i = 0; i < size; i++)
        din >> b.v[i];
    return (din);
}

ostream &operator<<(ostream &dout, vector &b)
{
    dout << "(" << b.v[0];
    for (int i = 1; i < size; i++)
        dout << "," << b.v[i];
    dout << ")";
    return (dout);
}

int x[size] = {2, 4, 6};
int main()
{
    vector m;
    vector n = x;

    cout << "Enter Elements of vector m"
         << "\n";
    cin >> m;
    cout << "\n";
    cout << "m=" << m << "\n";

    vector p, q;

    p = 2 * m;
    q = n * 2;

    cout << "\n";
    cout << "p=" << p << "\n";
    cout << "q=" << q << "\n";
```

```
    return 0;
}
```

```
// Write a program for developing a matrix class which can handle integer matrices of different
// dimensions. Also overload the operator for addition, multiplication & comparison of
// matrices.
#include <iostream>
using namespace std;
int n;
class matrix
{
    int n;
    int **a;

public:
    matrix(int nn)
    {
        n = nn;
        a = new int *[n];
        for (int i = 0; i < n; i++)
        {
            a[i] = new int[nn];
        }
    }
    friend matrix operator+(matrix m1, matrix m2);
    friend matrix operator*(matrix m1, matrix m2);
    friend bool operator==(matrix m1, matrix m2);
    void getMatrix()
    {
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < n; j++)
            {
                cin >> a[i][j];
            }
        }
    }
    void printMatrix()
    {
        for (int i = 0; i < n; i++)
        {
```

```
            for (int j = 0; j < n; j++)
            {
                cout << a[i][j] << " ";
            }
            cout << endl;
        }
    }
};

matrix operator+(matrix m1, matrix m2)
{
    matrix temp(3);
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            temp.a[i][j] = m1.a[i][j] + m2.a[i][j];
        }
    }
    return temp;
}

matrix operator*(matrix m1, matrix m2)
{
    matrix temp(3);
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            temp.a[i][j] = 0;
            for (int k = 0; k < n; k++)
            {
                temp.a[i][j] += m1.a[i][k] * m2.a[k][j];
            }
        }
    }
    return temp;
}

bool operator==(matrix m1, matrix m2)
{
    int flag = 33;
    for (int i = 0; i < n; i++)
    {
```

```
        if (flag == 33)
        {
            for (int j = 0; j < n; j++)
            {
                if (m1.a[i][j] == m2.a[i][j])
                {
                    flag = 33;
                }
                else
                {
                    flag = 44;
                    break;
                }
            }
        }
        else
        {
            break;
        }
    }
    if (flag == 33)
    {
        return true;
    }
    else
    {
        return false;
    }
}

int main()
{
    cin >> n;
    matrix m1(n), m2(n), m3(n), m4(n);

    m1.getMatrix();
    m2.getMatrix();

    m3 = m1 + m2;
    cout << "Addition of two given Matrices is : " << endl;
    m3.printMatrix();
    m4 = m1 * m2;
    cout << "Multiplication of two given Matrices is : " << endl;
    m4.printMatrix();
```

```
    bool a = m1 == m2;

    if (a)
    {
        cout << "The matrices are equal" << endl;
    }
    else
    {
        cout << "The matrices are not equal" << endl;
    }

    return 0;
}
```

```
// Write a program to overload new/delete operators in a class.
#include<iostream>
#include<string.h>
#include<new>
#include<stdlib.h>

using namespace std;
const int MAX = 5;
const int FREE = 0;
const int OCCUPIED = 1;

void memwarning( )
{
        cout << endl <<"Free store has now gone empty";
        exit( 1 );
}

class employee
{
        private:
                char name[20];
                int age;
                float sal;
        public:
                void *operator new(size_t bytes);
                void operator delete( void * q );
```

```
                void setdata( char * n, int a, float s );
                void showdata();
                        ~employee();
};

struct pool
{
        employee obj;
        int status;
};

int flag=0;
struct pool *p = NULL;

void * employee::operator new( size_t sz )
{
        int i;
        if( flag == 0 )
        {
                p = ( pool * )malloc( sz * MAX );
                if( p == NULL )
                        memwarning( );
                for( i = 0; i < MAX; i++ )
                        p[ i ].status = FREE;
                flag = 1;
                p[ 0 ].status = OCCUPIED;
                return &p[ 0 ].obj;
        }
        else
        {
                for( i = 0; i < MAX; i++ )
                {
                        if( p[ i ].status = FREE )
                        {
                                p[ i ].status = OCCUPIED;
                                return &p[ i ].obj;
                        }
                }
                memwarning( );
        }
}

void employee::operator delete( void * q )
{
```

```
        if(q == NULL)
                return;
        for(int i = 0; i < MAX; i++)
        {
                if(q == &p[ i ].obj)
                {
                        p[i].status = FREE;
                        strcpy(p[i].obj.name, "" );
                        p[i].obj.age = 0;
                        p[i].obj.sal = 0.0;
                }
        }
}

void employee::setdata( char * n, int a, float s )
{
        strcpy(name, n);
        age = a;
        sal = s;
}

void employee::showdata()
{
        cout << endl << name << "\t" << age << "\t" << sal;
}

employee::~employee()
{
        cout << endl << "reached destructor";
        free(p);
}

int main()
{
        void memwarning();
        set_new_handler(memwarning);
        employee * e1,*e2,*e3,*e4,*e5,*e6;
        e1 = new employee;
        e1->setdata("ajay", 23, 4500.50 );
        e2 = new employee;
        e2->setdata("amol", 25, 5500.50 );
        e3 = new employee;
        e3->setdata("anil", 26, 3500.50 );
        e4 = new employee;
```

```
        e4->setdata("anuj", 30, 6500.50 );
        e5 = new employee;
        e5->setdata("atul", 23, 4200.50 );
        e1->showdata();
        e2->showdata();
        e3->showdata();
        e4->showdata();
        e5->showdata();

        delete e4;
        delete e5;

        e4->showdata( );
        e5->showdata( );

        e4 = new employee;
        e5 = new employee;
        e6 = new employee;

        cout << endl << "Done!!";
        return 0;
}
```

```
// Write a program to maintain the records of person with details (Name and Age) and find the
// eldest among them. The program must use this pointer to return the result.
// INPUTS-
// John
// 23
// Vimal
// 24
#include <iostream>
using namespace std;
class Person
{
   char Name[20];
   int Age;

public:
   void getData()
   {
      // cout<<"Enter Person Name:";
```

```
        // gets(Name);
        cin >> Name;
        // cout<<"Enter Person Age:";
        cin >> Age;
    }
    void putData()
    {
        // cout<<"\n\nDetails About Eldest Person as follows:";
        cout << "Name = " << Name << endl;
        cout << "Age  = " << Age << endl;
    }
    Person &Compare(Person &p1)
    {
        if (p1.Age > this->Age)
            return p1;
        return *this;
    }
};
int main()
{
    Person x, y, z;
    x.getData();
    y.getData();
    z = x.Compare(y);
    z.putData();
    return 0;
}
```

```
// Write a Program to illustrate the use of pointers to objects which are related by inheritance.
// INPUTS-
// 100
// 200
#include <iostream>
using namespace std;
class BC
{
public:
    int b;
    void show()
    {
        cout << "b = " << b << endl;
```

```
    }
};
class DC : public BC
{
public:
    int d;
    void show()
    {
        cout << "b = " << b << endl;
        cout << "d = " << d << endl;
    }
};
int main()
{
    BC *bptr;
    BC base;
    int a;
    cin >> a;
    bptr = &base;
    bptr->b = a;
    cout << "bptr points to base object" << endl;
    bptr->show();
    DC derived;
    bptr = &derived;
    int der;
    cin >> der;
    bptr->b = der;
    cout << "bptr now points to derived object" << endl;
    bptr->show();
    return 0;
}
```

// Write a program to show conversion from string to int and vice-versa.

```
#include<iostream>
#include<sstream>
using namespace std;
int main()
{
    int k;
```

```
    cin>>k;
    stringstream ss;
    ss<<k;
    string s;
    cin>>s;
    ss>>s;
    string num;
    ss>>num;
    int i;
    ss>>i;
    cout<<"An integer value is: "<<k<<"\n";
    cout<<"Strig representation of an integer value is: "<<s<<endl;
    cout<<"The value of the string is: "<<s<<"\n";
    cout<<"Integer value the string is: "<<i;

    return 0;
}
```

Write a program to implement the exception handling.

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    string mobile;
    cin>>mobile;
    cout<<mobile<<endl;
    try
    {
        if(mobile.size()<10)
        {
            throw(mobile);
        }
        else
        {
            cout<<"The entered mobile number is valid";
        }
    }
    catch(string s)
    {
        cout<<"The mobile number should have 10 digits";
```

```
    }
}
```

Write a program to maintain the elementary database of employee using file concepts.

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int a;
    cin>>a;
    try{
        if(a==1)
        throw a;
        else if(a==2)
        throw 'A';
        else if(a==3)
        throw 4.5;
    }
    catch(int a)
    {
        cout<<"Integer exception caught."<<endl;
    }
    catch(char ch)
    {
        cout<<"Character exception caught."<<endl;
    }
    catch(double d)
    {
        cout<<"Double exception caught."<<endl;
    }
}
```

Write a program showing data conversion between objects of different classes and conversion routine should reside in destination class

```
#include<iostream>
using namespace std;
class stock2;
class stock1
{
int code, item;
float price;
public:
stock1(int a, int b, float c)
{
code=a;
item=b;
price=c;
}
void disp()
{
cout<<"code "<<code <<"\n";
cout<<"Items "<<item <<"\n";
cout<<"Price per item Rs . "<<price <<"\n";
}
int getcode()
{
 return code;
}
int getitem()
{
 return item;

}
int getprice()
{
 return price;

}
operator float()
{
return ( item*price );
}
};
```

```
class stock2
{
int code;
float val;
public:
stock2()
{
code=0; val=0;
}
stock2(int x, float y)
{
code=x; val=y;
}
void disp()
{
cout<< "code "<<code <<"\n";
cout<< "Total Value Rs . " <<val <<"\n";
}
stock2(stock1 p)
{
code=p.getcode();
val=p.getitem() * p.getprice();
}
};
int main ( )
{
 int a,b,c;
 cin>>a>>b>>c;
stock1 i1(a,b,c);//101,10,125.0
stock2 i2;
float tot_val;
tot_val=i1;
i2=i1;
cout<<"Stock Details-stock1-type " <<"\n";
i1.disp();
cout<<"Stock value"<<"\n";
cout<< tot_val<<"\n";
cout<<"Stock Details-stock2-type "<<"\n";
i2.disp();
}
```

Write a program to copy the contents of one file to another.

```
#include <iostream>
#include <fstream>
using std::ofstream;
using std::ifstream;
using namespace std;

int main()
{
    fstream file;
file.open("sample.txt",ios::out);
 ofstream outdata;
 int count=0;
 string str="Today C++ is the most widely used System Programming Language.\nMost of the
state of the art software have been implemented using C++.\nEasy to learn\nStructured
language\nIt produces efficient programs.\nIt can handle low-level activities.\nIt can be compiled
on a variety of computers.";
 string line;

 outdata.open("sample.txt");
 outdata << str << endl;
 outdata.close();
 cout<<"File content:\n";

 while(!file.eof())
 {
 file>>str;
 cout<<str;
 }
 /*ifstream ifile("sample.txt");
 while (getline(ifile, line))
 {
 count++;
 }
 cout << "\nNumbers of lines in the file : " << count << endl;*/
file.close();
    //ifile.close();
```

```
   return 0 ;
}
```

Write a program showing data conversion between objects of different classes.

```
#include<iostream>
using namespace std;

class Time
{
   private:
   int hours;
   int mins;
   public:
   Time()
   {
      hours=0;
      mins=0;
   }
   Time(int t)
   {
      hours=t/60;
      mins=t%60;
   }
   void display()
   {
      cout<<"Time = "<<hours<<" hrs and "<<mins<<" mins"<<endl;
   }
};

int main()
{
   Time T1;
   int dur;
   cin>>dur;
   T1=dur;
   T1.display();
   return 0;
}
```

// Write a program to show conversion from string to int and vice-versa.

```
#include<iostream>
#include<sstream>
using namespace std;
int main()
{
   int k;
   cin>>k;
   stringstream ss;
   ss<<k;
   string s;
   cin>>s;
   ss>>s;
   string num;
   ss>>num;
   int i;
   ss>>i;
   cout<<"An integer value is: "<<k<<"\n";
   cout<<"Strig representation of an integer value is: "<<s<<endl;
   cout<<"The value of the string is: "<<s<<"\n";
   cout<<"Integer value the string is: "<<i;

   return 0;
}
```

Write a program to design a class representing the information regarding digital library (books, tape: book & tape should be separate classes having the base class as media). The class should have the functionality for adding new item, issuing, deposit etc. the program should use the runtime polymorphism.

```
#include<iostream>
#include<cstring>
using namespace std;
class media
{
```

```
 protected:
 char title[50];
 float price;
 public:
 media(char *s, float a)
 {
 strcpy(title, s);
 price = a;
 }
 virtual void display()
 {

 }
};
class book : public media
{
 int pages;
 public:
 book(char *s, float a, int p) : media(s,a)
 {
 pages = p;
 }
 void display();
};
class tape : public media
{
 float time;
 public:
 tape(char * s, float a, float t):media(s,a)
 {
 time =t;
 }
 void display();
};
 void book ::display()
 {
 cout<<"\n Title:"<<title;
 cout<<"\n Pages:"<<pages;
 cout<<"\n Price:"<<price;
 }
 void tape ::display ()
 {
 cout<<"\n Title:"<<title;
 cout<<"\n Play Time:"<<time<<"mins";
```

```
 cout<<"\n Price:"<<price;
 }
 int main()
 {
 char * title = new char[30];
 float price, time;
 int pages;
 //cout<<"\n Enter Book Details \n"; cout<<"\n Title:";
 cin>>title;
 //cout<<"\n Price:";
 cin>>price;
 //cout<<"\n Pages:";
 cin>>pages;
 book book1(title, price, pages);
 // cout<<"\n Enter Tape Details";
 // cout<<"\n Title:";
 cin>>title;
 //cout<<"\n Price:";
 cin>>price;
 //cout<<"\n Play Times(mins):";
 cin>>time;

 tape tape1(title, price, time); media* list[2];
 list[0] = &book1;
 list[1] = &tape1;
 cout<<"Media Details";
 cout<<"\n.............Book.        ";
 list[0]->display ();

 cout<<"\n.............Tape.        ";
 list[1]->display ();
return 0;
}
```

Write a program illustrating the use of virtual functions in class.

```
#include <iostream>
using namespace std;
class Base
{
	public:
	virtual void print()
	{
		cout<<"Base Function"<<endl;
	}
};
class derived : public Base{
	public:
	void print() {
		cout << "bptr points to Base" << endl
		<< "Display Base" <<endl
		<<" Show Base:"<<endl
		<<" bptr points to derived"<<endl
		<<"Display Base"<<endl
		<<" Show Derived"<<endl
	}
};
int main() {
	derived derived1;
	Base* base1 = &derived1;
	base1->print();
	return 0;
}
```

Write a Program illustrating how the constructors are implemented and the order in which they are called when the classes are inherited. Use three classes Named alpha, beta, gamma such that alpha, beta are base class and gamma is derived class inheriting alpha & beta

```
#include <iostream>
using namespace std;

class alpha
{
   int x;

public:
   alpha(int i)
   {
      x = i;
      cout << "alpha initialiendl"<<endl;
   }

   void show_x(void)
   {
      cout << "x=" << x <<endl";
   }
};

class beta
{
   float y;

public:
   beta(float j)
   {
      y = j;
      cout << "beta initialized"<<endl;
   }

   void show_y(void)
   {
      cout << "y= " << y <<endl;
   }
};
```

```
class gamma : public beta, public alpha
{
    int m, n;

public:
    gamma(int a, float b, int c, int d) : alpha(a), beta(b)
    {
        m = c;
        n = d;
        cout <<"gamma initialized"<<endl;
    }

    void show_mn(void)
    {
        cout << "m=" << m <<endl;
        cout << "n=" << n <<endl;
    }
};
int main()
{
    int a, c, d;
    cin >> a;
    float b;
    cin >> b;
    cin >> c >> d;
    gamma g(a, b, c, d);
    g.show_x();
    g.show_y();
    g.show_mn();
    return 0;
}
```

Write a Program to design a student class representing student roll no. and a test class (derived class of student) representing the scores of the student in various subjects and sports class representing the score in sports. The sports and test class should be inherited by a result class having the functionality to add the scores and display the final result for a student

```
#include<iostream>
using namespace std;

class Student
{
   protected:
   int roll_number;
   public:
   void get_number(int a)
   {
      roll_number = a;
   }
   void put_number(void)
   {
      cout<<"Roll No:"<<roll_number<<"\n";
   }
};

class Test : 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
{
```

```
    protected:
    float score;
    public:
    void get_score(float s)
    {
        score = s;
     }
    void put_score(void)
    {
        cout<<"Sports wt:"<<score<<"\n\n";
    }
};

class Result : public Test, public Sports
{
    float total;
    public:
    void display(void);
};

void Result ::display(void)
{
    total = part1 + part2 + score;
    put_number();
    put_marks();
    put_score();
    cout<<"Total Score:"<<total<<"\n";
}

int main()
{
    Result student_1;
    int num;
    float m1,m2;
    float s;
    cin>>num>>m1>>m2>>s;
    student_1.get_number (num);
    student_1.get_marks (m1, m2);
    student_1.get_score (s);
    student_1.display ();
    return 0;
}
```

Write a program to overload new/delete operators in a class

```
// Write a program to overload new/delete operators in a class.
#include<iostream>
#include<string.h>
#include<new>
#include<stdlib.h>

using namespace std;
const int MAX = 5;
const int FREE = 0;
const int OCCUPIED = 1;

void memwarning( )
{
	cout << endl <<"Free store has now gone empty";
	exit( 1 );
}

class employee
{
	private:
		char name[20];
		int age;
		float sal;
	public:
		void *operator new(size_t bytes);
		void operator delete( void * q );
		void setdata( char * n, int a, float s );
		void showdata();
			~employee();
};

struct pool
{
	employee obj;
	int status;
};

int flag=0;
struct pool *p = NULL;
```

```
void * employee::operator new( size_t sz )
{
	int i;
	if( flag == 0 )
	{
		p = ( pool * )malloc( sz * MAX );
		if( p == NULL )
			memwarning( );
		for( i = 0; i < MAX; i++ )
			p[ i ].status = FREE;
		flag = 1;
		p[ 0 ].status = OCCUPIED;
		return &p[ 0 ].obj;
	}
	else
	{
		for( i = 0; i < MAX; i++ )
		{
			if( p[ i ].status = FREE )
			{
				p[ i ].status = OCCUPIED;
				return &p[ i ].obj;
			}
		}
		memwarning( );
	}
}

void employee::operator delete( void * q )
{
	if(q == NULL)
		return;
	for(int i = 0; i < MAX; i++)
	{
		if(q == &p[ i ].obj)
		{
			p[i].status = FREE;
			strcpy(p[i].obj.name, "" );
			p[i].obj.age = 0;
			p[i].obj.sal = 0.0;
		}
	}
}
```

```
void employee::setdata( char * n, int a, float s )
{
	strcpy(name, n);
	age = a;
	sal = s;
}

void employee::showdata()
{
	cout << endl << name << "\t" << age << "\t" << sal;
}

employee::~employee()
{
	cout << endl << "reached destructor";
	free(p);
}

int main()
{
	void memwarning();
	set_new_handler(memwarning);
	employee * e1,*e2,*e3,*e4,*e5,*e6;
	e1 = new employee;
	e1->setdata("ajay", 23, 4500.50 );
	e2 = new employee;
	e2->setdata("amol", 25, 5500.50 );
	e3 = new employee;
	e3->setdata("anil", 26, 3500.50 );
	e4 = new employee;
	e4->setdata("anuj", 30, 6500.50 );
	e5 = new employee;
	e5->setdata("atul", 23, 4200.50 );
	e1->showdata();
	e2->showdata();
	e3->showdata();
	e4->showdata();
	e5->showdata();

	delete e4;
	delete e5;

	e4->showdata( );
```

```
        e5->showdata( );

        e4 = new employee;
        e5 = new employee;
        e6 = new employee;

        cout << endl << "Done!!";
        return 0;
}
```

Write a program in C++ to highlight the difference between an overloaded assignment operator and copy construct.