TABLE OF CONTENTS

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.h>
#include<conio.h>
class test
{
      int code;
      static int count;

public:
      void setcode(void)
      {
            code = ++count;
      }
      void showcode(void)
      {
            cout<<"object number:"<<code<<"\n";
      }
      static void showcount(void)
      {
            cout<<"count:"<<count<<"\n";
      }
};

int test :: count;
int main()
{
      test t1,t2;

      t1.setcode();
      t2.setcode();

      test :: showcount();

      test t3;
      t3.setcode();

      test :: showcount();
      t1.showcode();
      t2.showcode();
      t3.showcode();
      return 0;
}
```

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.h>
const 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];
      count++;
}
void ITEMS :: displaySum(void)
{
      float sum=0;
      for(int i=0;i<count;i++)
            sum=sum+itemPrice[i];
      cout<<"\n Total Value:"<<sum<<"\n";
}
void ITEMS :: remove(void)
{
      int a;
      cout<<"Enter Item Code";
      cin>>a;
      for(int i=0;i<count;i++)
            if(itemCode[i] == a)
                  itemPrice[i]=0;
}
void ITEMS :: displayItems(void)
{
      cout<<"\n Code    Price\n";

      for(int i=0;i<count;i++)
      {
            cout<<"\n"<<itemCode[i];
            cout<<"     "<<itemPrice[i];
      }
      cout<<"\n";
}



int main()
```

```
{
      ITEMS order;
      order.CNT();
      int x;
      do
      {
            cout<<"\n You can do the following;"
                  <<"Enter appropriate number\n";
            cout<<"\n1 : Add an Item";
            cout<<"\n2 : Display Total Value";
            cout<<"\n3 : Delete an Item";
            cout<<"\n4 : Display all items";
            cout<<"\n5 : Quit";
            cout<<"\n\n What 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;
            default : cout<<"Error in input";
            }
      }while(x!=5);
      return 0;
}
```

3. 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.h>
#include<conio.h>
class employee
{
      char name [30];
      float age;
public:
      void getdata(void);
      void putdata(void);
};
void employee :: getdata(void)
{
      cout<<"Enter Name";
      cin>>name;
      cout<<"Enter Age";
      cin>>age;
}
void employee :: putdata(void)
{
      cout<<"Name:"<<name<<"\n";
      cout<<"Age:   "<<age<<"\n";
}
const int size=3;
int main()
{

      employee manager[size];
      for(int i=0; i<size; i++)
      {
            cout<<"\n Details of manager"<<i+1<<"\n";
            manager[i].getdata();
      }
      cout<<"\n";
      for(i=0; i<size; i++)
      {
            cout<<"\n Manager"<<i+1<<"\n";
            manager[i].putdata();
      }

      return 0;
}
```

4. 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.h>
#include<conio.h>
class ABC;
class XYZ
{
      int x;
public:
      void setvalue(int i)
      {
            x=i;
       }
      friend void max(XYZ, ABC);
};
class ABC
{
      int a;
public:
      void setvalue(int i)
      {
            a=i;
      }
      friend void max(XYZ, ABC);
};
void max (XYZ m, ABC n)
{
      if(m.x>=n.a)
            cout<<m.x;
      else
            cout<<n.a;
}
int main()
{
      ABC abc;
      abc.setvalue(10);
      XYZ xyz;
      xyz.setvalue(20);
      max(xyz,abc);

      return 0;
}
```

5. Write a Program to swap private data members of classes named as *class_1, class_2* using friend function.

```
#include<iostream.h>
#include<conio.h>
class class_2;
class class_1
{
      int value1;
public:
      void indata(int a)
      {
            value1=a;
      }
      void display(void)
      {
            cout<<value1<<"\n";
      }
      friend void exchange(class_1 &, class_2 &);
};
class class_2
{
      int value2;
public:
      void indata(int a)
      {
            value2=a;
      }
      void display(void)
      {
            cout<<value2<<"\n";
      }
      friend void exchange(class_1 &, class_2 &);
};
void exchange(class_1 &x, class_2 &y)
{
      int temp = x.value1;
      x.value1 = y.value2;
      y.value2 = temp;
}
int main()
{
      class_1 C1;
      class_2 C2;

      C1.indata(100);
      C2.indata(200);
      cout<<"Values before exchange"<<"\n";

      C1.display();
      C2.display();
      exchange(C1, C2);
      cout<<"Values after exchange"<<"\n";
      C1.display();
      C2.display();
      return 0;
}
```

6. Write a Program to design a class complex to represent complex numbers. The complex class shuold 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.h>
#include<conio.h>
class complex
{
      float x;
      float y;
public:
      void input(float real, float img)
      {
            x=real;
            y=img;
      }
      friend complex sum(complex, complex);
      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 complex :: show(complex c)
{
      cout<<c.x<<"+j"<<c.y<<"\n";
}
int main()
{
      complex A,B,C;
      A.input(3.1, 5.65);
      B.input(2.75, 1.2);

      C=sum(A,B);
      cout<<"A=";
      A.show(A);
      cout<<"B=";
      B.show(B);
      cout<<"C=";
      C.show(C);

      return 0;
}
```

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

```
#include<iostream.h>
#include<conio.h>
class code
{
      int id;
public:
      code(){}
      code(int a)
      {
            id = a;
      }
      code(code & x)
      {
            id = x.id;
      }
      void display(void)
      {
            cout<<id;
      }
};
int main()
{

      code A(100);
      code B(A);
      code C = A;
      code D;
      D = A;
      cout<<"\n id of A:";
      A.display();
      cout<<"\n id of B:";
      B.display();
      cout<<"\n id of C:";
      C.display();
      cout<<"\n id of D:";
      D.display();

      return 0;
}
```

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

```
#include<iostream.h>
#include<string.h>
#include<conio.h>
class String
{
      char *name;
      int length;
public:
      String()
      {
            length = 0;
            name = new char[length +1];
      }
      String (char *s)
      {
            length = strlen(s);
            name= new char[length + 1];
            strcpy(name, s);
      }
      void display(void)
      {
            cout<<name<<"\n";
      }
      void join(String &a, String &b);
};

void String :: join (String &a, String &b)
{
      length = a.length + b.length;
      delete name;
      name = new char [length + 1];

      strcpy(name,a.name);
      strcat(name, b.name);
};

int main()
{

      char *first = "Joseph";
      String name1(first), name2("Louis "), name3("Lagrange"),s1,s2;
      s1.join(name1, name2);
      s2.join(s1, name3);
      name1.display();
      name2.display();
      name3.display();
      s1.display();
      s2.display();

      return 0;
}
```

9. 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.h>

class matrix
{
      int **p;
      int 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 size of matrix";
      cin>>m>>n;
      matrix A(m,n);
      cout<<"Enter Matrix Element row by row:";
      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<<"\n";
      cout<<A.put_element(1,2);
      return 0;
}
```

10. 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.h>
class complex
{
private:
    float real,
           imag;
public:
    complex( )
    {
    }
    complex( float  r, float  i )
    {
        real = r;
        imag = i;
    }

    void getdata( )
    {
        float r,
               i;
        cout << endl << "Enter real and imaginary part ";
        cin >> r >> i;
        real = r;
        imag = i;
    }

    void setdata( )
    {
        real = r;
        imag = i;
    }

    void displaydata( )
    {
        cout << endl << "real = " << real;
        cout<<endl<<Imaginary = "<<imag;
    }

    complex  operator +( complex  c )
    {
        complex  t;
        t.real = real + c.real;
        t.imag = imag + c.imag;
    }

    complex  operator *( complex  c )
    {
        complex  t;
        t.real = real * c.real - imag * c.imag;
        t.imag = real * c.imag + c.real * imag;
        return t;
    }
}    ;
void main( )
{
```

```
    complex  c1,
             c2 ( 1.2, -2.5 ),
              c3,
              c4;
    c1.setdata( 2.0, 2.0 );
    c3 = c1 + c2;
    c3.displaydata( );
    c4.getdata( );
    complex  c5 ( 2.5, 3.0 ),
              c6;
    c6 = c4 * c5;
    c6.displaydata( );
    complex  c7;
    c7 = c1 + c2 * c3;
    c7.displaydata( );
}
```

11. Write a Program to overload operators like *, <<, >> using friend function. The following overloaded operators should work for a class *vector*.

```
#include<iostream.h>
#include<conio.h>

const 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;
}
```

12. 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.h>
#include     <iomanip.h>
class matrix
{
    int  maxrow, maxcol;
    int * ptr;
public:
    matrix( int  r, int  c )
    {
        maxrow = r;
        maxcol = c;
        ptr    = new int [r * c];
    }
    void getmat( )
    {
        int i,j, mat_off,temp;
        cout << endl << "enter elements matrix:" << endl;
        for( i = 0; i < maxrow; i++ )
        {
            for( j = 0; j < maxcol; j++ )
            {
                mat_off = i * maxcol + j;
                cin >> ptr[ mat_off ];
            }
        }
    }
    void printmat( )
    {
        int i, j, mat_off;
        for( i = 0; i < maxrow; i++ )
        {
            cout << endl;
            for( j = 0; j < maxcol; j++ )
            {
                mat_off = i * maxcol + j;
                cout << setw( 3 ) << ptr[ mat_off ];
            }
        }
    }
    int delmat( )
    {
        matrix  q ( maxrow - 1, maxcol - 1 );
        int     sign  = 1, sum  = 0, i, j,k,count;
        int     newsize,newpos,pos,order;
        order = maxrow;
        if( order == 1 )
        {
            return ( ptr[ 0 ] );
        }
        for( i = 0; i < order; i++, sign *= -1 )
        {
            for( j = 1; j < order; j++ )
            {
                for( k = 0, count = 0; k < order;
                     k++ )
```

```
            {
                if( k == i )
                    continue;
                pos = j * order + k;
                newpos = ( j - 1 ) * ( order - 1 ) + count;
                q.ptr[ newpos ] = ptr[ pos ];
                count++;
            }
        }
        sum = sum + ptr[ i ] * sign * q.delmat( );
    }
    return ( sum );
}
matrix  operator +( matrix  b )
{
    matrix  c ( maxrow, maxcol );
    int     i,j,mat_off;
    for( i = 0; i < maxrow; i++ )
    {
        for( j = 0; j < maxcol; j++ )
        {
            mat_off = i * maxcol + j;
            c.ptr[ mat_off ] = ptr[ mat_off ] + b.ptr[ mat_off ];
        }
    }
    return ( c );
}
matrix  operator *( matrix  b )
{
    matrix  c ( b.maxcol, maxrow );
    int     i,j,k,mat_off1, mat_off2, mat_off3;
    for( i = 0; i < c.maxrow; i++ )
    {
        for( j = 0; j < c.maxcol; j++ )
        {
            mat_off3 - i * c.maxcol + j;
            c.ptr[ mat_off3 ] = 0;
            for( k = 0; k < b.maxrow; k++ )
            {
                mat_off2 = k * b.maxcol + j;
                mat_off1 = i * maxcol + k;
                c.ptr[mat_off3]+=ptr[mat_off1]* b.ptr[mat_off2 ];
            }
        }
    }
    return ( c );
}
int operator ==( matrix  b )
{
    int i,j, mat_off;
    if( maxrow != b.maxrow
         || maxcol != b.maxcol )
        return ( 0 );
    for( i = 0; i < maxrow; i++ )
    {
        for( j = 0; j < maxcol; j++ )
        {
            mat_off = i * maxcol + j;
            if( ptr[ mat_off ]
```

```
                    != b.ptr[ mat_off ] )
                   return ( 0 );
            }
        }
        return ( 1 );
    }
}    ;

void main( )
{
    int rowa, cola, rowb, colb;
    cout << endl << "Enter dimensions of matrix A ";
    cin >> rowa >> cola;
    matrix  a ( rowa, cola );
    a.getmat( );
    cout << endl << "Enter dimensions of matrix B";
    cin >> rowb >> colb;
    matrix  b ( rowb, colb );
    b.getmat( );
    matrix  c ( rowa, cola );
    c = a + b;
    cout << endl << "The sum of two matrics = ";
    c.printmat( );
    matrix  d ( rowa, colb );
    d = a * b;
    cout << endl << "The product of two matrics = ";
    d.printmat( );
    cout << endl << "Determinant of matrix a =" << a.delmat( );
    if( a == b )
        cout << endl << "a & b are equal";
    else
        cout << endl << "a & b are not equal";
}
```

13. Write a program to overload *new/delete* operators in a class.

```
#include     <iostream.h>
#include     <stdlib.h>
#include     <string.h>
#include     <new.h>

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 );
}

void 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!!";
}
```

14. Write a program in C++ to highlight the difference between overloaded *assignment operator* and *copy constructor*.

```
#include     <iostream.h>

class circle
{
private:
    int   radius;
    float x, y;

public:
    circle( )
    {
    }
    circle( int  rr, float  xx, float  yy )
    {
        radius = rr;
        x      = xx;
        y      = yy;
    }
    circle  operator =( circle & c )
    {
        cout << endl << "Assignment operator invoked";
        radiius = c.radius;
        x       = c.x;
        y       = c.y;
        return circle( radius, x, y );
    }
    circle( circle & c )
    {
        cout << endl << "copy constructor invoked";
        radius = c.radius;
        x      = c.x;
        y      = c.y;
    }
    void showdata( )
    {
        cout << endl << "Radius = " << radius;
        cout << endl << "X-Coordinate=" << x;
        cout << endl << "Y-Coordinate=" << y;
    }
}    ;

void main( )
{
    circle  c1 ( 10, 2.5, 2.5 );
    circle  c2,c4;
    c4 = c2 = c1;
    circle  c3  = c1;
    c1.showdata( );
    c2.showdata( );
    c3.showdata( );
    c4.showdata( );
}
```

15. 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.h>
#include<conio.h>
class alpha
{
      int x;
public:
      alpha(int i)
      {
            x = i;
            cout<<"alpha initialized\n";
      }
      void show_x(void)
      {
            cout<<"x="<<x<<"\n";
      }
};
class beta
{
      float y;
public:
      beta(float j)
      {
            y=j;
            cout<<"beta initialized\n";
      }
      void show_y(void)
      {
            cout<<"y= "<<y<<"\n";
      }
};
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\n";
      }
      void show_mn(void){
            cout<<"m="<<m<<"\n";
            cout<<"n="<<n<<"\n";
      }
};
void main()
{
      gamma g(5, 10.75, 20, 30);
      g.show_x();
      g.show_y();
      g.show_mn();
}
```

16. Write a Program to design a stuent 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.h>

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;

      student_1.get_number (1234);
      student_1.get_marks (27.5, 33.0);

      student_1.get_score (6.0);

      student_1.display ();

      return 0;
}
```

17. 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.

```
#include<iostream.h>
#include<string.h>

class person
{
      char name[20];
      float age;

public:
      person(char *s, float a)
      {
            strcpy(name, s);
            age = a;
      }

      person & person :: greater(person & x)
      {
            if(x.age >= age)
                  return x;
            else
                  return *this;
      }

      void display(void)
      {
            cout<<"Name:"<<name<<"\n"
                  <<"Age: "<<age<<"\n";
      }
};

int main()
{
      person p1("John", 37.50),
               p2("Ahmed",29.0),
               p3("Hebber", 40.5);

      person p = p1.greater (p3);
      cout<<"Elder Person is:\n";
      p.display();

      p = p1.greater (p2);
      cout<<"Elder Person is:\n";
      p.display();

      return 0;
}
```

18. Write a Program to illustrate the use of pointers to objects whch are related by inheritance.

```
#include<iostream.h>

class BC
{
public:
      int b;
      void show()
      {
            cout<<"b="<<b<<"\n";
      }
};

class DC : public BC
{
      public:
            int d;
            void show()
            {
                  cout<<"b="<<b<<"\n"
                        <<"d="<<d<<"\n";
            }
};

int main()
{
      BC *bptr;
      BC base;
      bptr = &base;

      bptr->b = 100;
      cout<<"bptr points to base object\n";
      bptr->show ();


      DC derived;
      bptr = &derived;
      bptr->b = 200;

      cout<<"bptr now points to derived object\n";
      bptr->show ();

      DC *dptr;
      dptr = &derived;
      dptr->d = 300;

      cout<<"dptr is derived type pointer\n";
      dptr->show ();

      cout<<"Using ((DC *)bptr)\n";
      ((DC *)bptr)->d = 400;
      ((DC *)bptr)->show ();

      return 0;
}
```

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

```
#include<iostream.h>

class Base
{
      public:
            void display()
            {
                  cout<<"\n Display Base";
            }

            virtual void show()
            {
                  cout<<"\n Show Base:";
            }
};

class Derived : public Base
{
      public:
            void display()
            {
                  cout<<"\n Display Derived";
            }

            void show()
            {
                  cout<<"\n Show Derived";
            }
};

int main()
{
      Base B;
      Derived D;
      Base *bptr;

      cout<<"\n bptr points to Base\n";
      bptr = &B;
      bptr ->display ();
      bptr ->show ();

      cout<<"\n\n bptr points to derived\n";
      bptr = &D;
      bptr ->display ();
      bptr ->show ();

      return 0;
}
```

20. 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.h>
#include<string.h>

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<<"\n Media Details";

	cout<<"\n..............Book.....";
	list[0]->display ();

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

	return 0;
}
```

21. write a program to show conversion from string to int and vice-versa.

```
#include     <iostream.h>
#include     <stdlib.h>
#include     <string.h>
class string
{
private:
    char str[ 20 ];
public:
    string( )
    {
        str[ 0 ] = '\0';
    }
    string( char * s )
    {
        strcpy( str, s );
    }
    string( int  a )
    {
        itoa( a, str, 10 );
    }
    operator int( )
    {
        int i  = 0,
             l,
             ss  = 0,
             k  = 1;
        l = strlen( str ) - 1;
        while( l >= 0 )
        {
            ss = ss + ( str[ l ] - 48 ) * k;
            l--;
            k *= 10;
        }
        return ( ss );
    }
    void displaydata( )
    {
        cout << str;
    }
}    ;
void main( )
{
    string  s1  = 123;
    cout << endl << "s1=";
    s1.displaydata( );
    s1 = 150;
    cout << endl << "s1=";
    s1.displaydata( );
    string  s2 ( "123" );
    int     i  = int( s2 );
    cout << endl << "i=" << i;
    string  s3 ( "456" );
    i = s3;
    cout << endl << "i=" << i;
}
```

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

```
#include     <iostream.h>
#include     <stdlib.h>
#include     <string.h>

class date
{
private:

    char dt[ 9 ];

public:

    date( )
    {
        dt[ 0 ] = '\0';
    }

    date( char * s )
    {
        strcpy( dt, s );
    }

    void displaydata( )
    {
        cout << dt;
    }

}    ;

class dmy
{
private:

    int day,
         mth,
         yr;
public:

    dmy( )
    {
        day = mth = yr = 0;
    }

    dmy( int  d, int  m, int  y )
    {
        day = d;
        mth = m;
        yr  = y;
    }    ;

    operator date( )
    {
        char temp[ 3 ], str[ 9 ];
        itoa( day, str, 10 );
        strcat( str, "/" );
        itoa( mth, temp, 10 );
```

```
        strcat( str, temp );
        strcat( str, "/" );
        itoa( yr, temp, 10 );
        strcat( str, temp );
        return ( date( str ) );
    }

    void displaydata( )
    {
        cout << day << "\t" << mth << "\t" << yr;
    }

}    ;

void main( )
{
    date  d1;
    dmy   d2 ( 17, 11, 94 );
    d1 = d2;

    cout<,endl<<"d1=";
    d1.displaydata( );

    cout << endl << "d2=";
    d2.displaydata( );

}
```

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

```
#include     <iostream.h>
#include     <string.h>
#include     <stdlib.h>

class dmy
{
    int day,
         mth,
         yr;
public:
    dmy( )
    {
        day = mth, yr = 0;
    }

    dmy( int  d, int  m, int  y )
    {
        day = d;
        mth = m;
        yr  = y;
    }

    int getday( )
    {
        return ( day );
    }

    int getmth( )
    {
        return ( mth );
    }

    int getyr( )
    {
        return ( yr );
    }

    void displaydata( )
    {
        cout << day << "\t" << mth << "\t" << yr;
    }
}    ;

class date
{
private:
    char dt[ 9 ];

public:
    date( )
    {
        dt[ 0 ] = '\0';
    }
    date( char * s )
    {
        strcpy( dt, s );
```

```
    }

    void displaydata( )
    {
        cout << dt;
    }

    date( dmy  t )
    {
        int  d  = t.getday( );
        int  m  = t.getmth( );
        int  y  = t.getyr( );
        char temp[ 3 ];
        itoa( d, dt, 10 );
        strcat( dt, "\t" );
        itoa( m, temp, 10 );
        strcat( dt, temp );
        strcat( dt, "/" );
        itoa( y, temp, 10 );
        strcat( dt, temp );
    }
}    ;

void main( )
{
    date  d1;
    dmy   d2 ( 17, 11, 94 );
    d1 = d2;
    cout << endl << "d1=";
    d1.displaydata( );
    cout << endl << "d2=";
    d2.displaydata( );
}
```

24. Write a program implementing basic operation of *class ios* i.e. setf,unsetf,precision etc.

```
#include     <iostream.h>
#include     <conio.h>

void main( )
{
    int   i  = 52;
    float a  = 425.0;
    float b  = 123.500328;

    char  str[ ]  = "Dream. Then make it happend!";
    clrscr( );

    cout.setf( ios::unitbuf );
    cout.setf( ios::stdio );
    cout.setf( ios::showpos );

    cout << i << endl;

    cout.setf( ios::showbase );
    cout.setf( ios::uppercase );
    cout.setf( ios::hex, ios::basefield );

    cout << i << endl;

    cout.setf( ios::oct, ios::basefield );
    cout << i << endl;

    cout.fill( '0' );
    cout << "Fill character " << cout.fill( ) << endl;

    cout.setf( ios::dec, ios::basefield );
    cout.width( 10 );
    cout << i << endl;

    cout << setf( ios::left, ios::adjustfield );
    cout.width( 10 );
    cout << i << endl;

    cout.setf( ios::internal, ios::adjustfield );
    cout.width( 10 );
    cout << endl;
    cout << endl;

    cout.width( 10 );
    cout << str << endl;

    cout.width( 40 );
    cout.setf( ios::left, ios::adjustfield );
    cout.width( 40 );

    cout << str << endl;
    cout.precision( 6 );
    cout << "Precision" << cout.precision( );

    cout.setf( ios::showpoint );
    cout.unsetf( ios::showpos );
```

```
    cout << endl << a;

    cout.unsetf( ios::showpoint );
    cout << endl << a;

    cout.setf( ios::fixed, ios::floatfield );
    cout << endl << b;

    cout.setf( ios::scientific, ios::floatfield );
    cout << endl << b;

    b = 5.375;
    cout.precision( 14 );

    cout.setf( ios::fixed, ios::floatfield );
    cout << endl << b;

    cout.setf( ios::scientific, ios::floatfield );
    cout << endl << b;

    cout.unsetf( ios::showpoint );
    cout.unsetf( ios::unitbuf );
    cout.unsetf( ios::stdio );

}
```

25. Write a program to implement I/O operations on characters. I/O operations includes inputing a string, Calculating length of the string, Storing the String in a file, fetching the stored characters from it, etc.

```
#include<iostream.h>
#include<fstream.h>
#include<string.h>

int main()
{
      char string[80];
      cout<<"Enter a String \n";
      cin>>string;
      int len = strlen(string);

      fstream file;
      file.open("TEXT", ios::in | ios::out);

      for(int i=0;i<len;i++)
            file.put(string[i]);

      file.seekg(0);

      char ch;
      while(file)
      {
            file.get(ch);
            cout<<ch;
      }
      return 0;
}
```

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

```
#include     <fstream.h>

void main( )
{
    char cource[ 67 ],
          target[ 67 ];
    char ch;

    cout << endl << "Enter source filename";
    cin >> source;

    cout << endl << "Enter target filename";
    cin >> target;

    ifstream  infile ( source );
    ofstream  outfile ( target );

    while( infile )
    {
        infile.get( ch );
        outfile.put( ch );
    }

}
```

27. Write a program to perform read/write binary I/O operation on a file (i.e. write the object of a structure/class to file).

```
#include     <fstream.h>

void main( )
{
    struct employee
     {
        char  name[ 20 ];
        int   age;
        float basic;
        float gross;
     }    ;

    employee  e;

    char      ch  = 'Y';
    ofstream  outfile;

    outfile.open( "EMPLOYEE.DAT", ios::out | ios::binary );

    while( ch == 'Y' )
    {
        cout << endl << "Enter a record";
        cin >> e.name >> e.age >> e.basic >> e.gross;
        outfile.write( ( char * )&e, sizeof( e ) );
        cout << endl << "Add Another Y/N";
        cin >> ch;
    }

    outfile.close( );
    ifstream  infile;
    infile.open( "EMPLOYEE.DAT", ios::in | ios::binary );

    while( infile.read( ( char * )&e, sizeof( e ) ) )
    {
        cout << endl << e.name << "\t" << e.age << "\t" << e.basic << "\t"
         << e.gross;
    }
}
```

28. Write a program to maintain a elementary database of employees using files.

```
#include     <fstream.h>
#include     <conio.h>
#include     <stdlib.h>
#include     <stdio.h>
#include     <string.h>
#include     <iomanip.h>

class group
{
private:
    struct person
     {
        char  flag;
        char  empcode[ 5 ];
        char  name[ 40 ];
        int   age;
        float sal;
     }    p;
    fstream  file;

public:
         group( );
    void addrec( );
    void listrec( );
    void modirec( );
    void delrec( );
    void recallrec( );
    void packrec( );
    void exit( );
}    ;

void main( )
{
    char   choice;
    group  g;

    do
    {
        clrscr( );
        gotoxy( 30, 10 );
        cout << "1. Add records";
        gotoxy( 30, 11 );
        cout << "2. List records";
        gotoxy( 30, 12 );
        cout << "3. Modify records";
        gotoxy( 30, 13 );
        cout << "4. Delete records";
        gotoxy( 30, 14 );
        cout << "5. Recall records";
        gotoxy( 30, 15 );
        cout << "6. Pack records";
        gotoxy( 30, 16 );
        cout << "0. Exit";
        gotoxy( 30, 18 );
        cout << "Your Choice ? ";
        cin >> choice;
```

```
        clrscr( );

        switch( choice )
        {
            case '1':
                g.addrec( );
                break;
            case '2':
                g.listrec( );
                break;
            case '3':
                g.modirec( );
                break;
            case '4':
                g.delrec( );
                break;
            case '5':
                g.recallrec( );
                break;
            case '6':
                g.packrec( );
                break;
            case '0':
                g.exit( );
                break;
        }
    } while( choice != 0 );
}

void group::group( )
{
    file.open( "emp.dat", ios::binary || ios::in || ios::out );
    if( !file )
    {
        cout << endl << "Unable to open file";
        exit( );
    }
}

void group::addrec( )
{
    char ch;
    file.seekp( 0L, ios::end );

    do
    {
        cout << endl << "Enter emp code, name, age & salary" << endl;
        cin >> p.empcode >> p.name >> p.age >> p.sal;
        p.flag = '';
        file.write( ( char * )&p, sizeof( p ) );
        cout << "Add another record? (Y/N)";
        cin >> ch;
    } while( ch == 'Y' || ch == 'Y' );

}


void group::listrec( )
{
```

```
    int j  = 0,a;
    file.seekg( 0L, ios::beg );

    while( file.read( ( char * )&p, sizeof( p ) ) )
    {
        if( p.flag != '*' )
        {
            cout <<endl << "Record#" << j++ << setw( 6 )<< p.empcode
             <<setw(20)<<p.name<<setw(4<<p.age<<setw(9)<< p.sal;
        }

        file.clear( );
        cout << endl << "Press any key......";
        getch( );
    }

void group::modirec( )
{
          char code[ 5 ];
          int  count  = 0;
    long  int  pos;

    cout << "Enter employee code: ";
    cin >> code;

    file.seekg( 0L, ios::beg );
    while( file.read( ( char * )&p, sizeof( p ) ) )
    {
        if( strcmp( p.empcode, code ) == 0 )
        {
            cout << endl << "Enter new record" << endl;
            cin >> p.empcode >> p.name >> p.age;
            p.flag = '';
            pos    = count * sizeof( p );
            file.seekp( pos, ios::beg );
            file.write( ( char * )&p, sizeof( p ) );
            return;
        }

        count++;
    }
    cout << endl << "No employee in file with code = " << code;
    cout << endl << "Press any key .....";
    getch( );

    file.clear( );
}

void group::delrec( )
{
          char code[ 5 ];
            long  int  pos;
          int  count  = 0;
    cout << "Enter employee code : ";
    cin >> code;
    file.seekg( 0L, ios::beg );

    while( file.read( ( char * )&p, sizeof( p ) ) )
    {
```

```
        if( strcmp( p.empcode, code ) == 0 )
        {
            p.flag = '*';
            pos    = count * sizeof( p );
            file.seekp( pos, ios::beg );
            file.write( ( char * )&p, sizeof( p ) );
            return;
        }
        count++;
    }
    cout << endl << "No employee in file with code = " << code;
    cout<<endl<<Press any key ....";
    getch( );
    file.clear( );
}
void group.recallrec()
{
          char code[ 5 ];
    long  int  pos;
          int  count  = 0;
    cout << "Enter employee code: ";
    cin >> code;
    file.seekg( 0L, ios::beg );
    while( file.read( ( char * )&p, sizeof( p ) ) )
    {
        if( strcmp( p.empcode, code ) == 0 )
        {
            p.flag = '';
            pos    = count * sizeof( p );
            file.seekp( pos, ios::beg );
            file.write( ( char * )&p, sizeof( p ) );
            return;
        }
        count++;
    }
    cout << endl << "No employee in file with code = " << code;
    cout << endl << "Press any key ....";
    file.clear( );
void group::packrec( )
{
    ofstream  outfile;
    outfile.open( "TEMP", ios::out );
    file.seekg( 0, ios::beg );
    while( file.read( ( char * )&p, sizeof( p ) ) )
    {
        if( p.flag != '*' )
            outfile.write((char *)&p,sizeof(p)));
    }
    outfile.close( );    file.close( );
    remove( "EMP.dat" );
    rename( "TEMP", "TEMP.dat" );
    file.open( "EMP.dat", ios::binary | ios::in | ios::out | ios::nocreate );
}
void group::exit( )
{
    file.close( );
}
```

29. Write a Program for reading and writing data to and from the file using command line arguments.

```
#include<iostream.h>
#include<fstream.h>
#include<stdlib.h>

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]<<" ";
            else
                  fout1<<number[i]<<" ";
      }

      fout1.close();
      fout2.close();

      ifstream fin;

      char ch;

      for(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;
}
```

Note: - project->settings->debug->program arguments (specify the arguments to be passed)

30. Write a program showing implementation of stack class having the functionality of push, pop operations.

```
#include     <iostream.h>
#define     MAX     10
class stack
{
private:
    int arr[ MAX ], top;
public:
    stack( )
    {
        top = -1;
    }
    void push( int  item )
    {
        if( top == MAX - 1 )
        {
            cout << endl << "Stack is full";
            return;
        }
        top++;
        arr[ top ] = item;
    }
    int pop( )
    {
        if( top == -1 )
        {
            cout << endl << "Stack is empty";
            return NULL;
        }
        int data  = arr[ top ];
        top--;
        return data;
    }
}    ;
void main( )
{
    stack  s;
    s.push( 11 );
    s.push( 12 );
    s.push( 13 );
    s.push( 14 );
    s.push( 15 );
    s.push( 16 );
    s.push( 17 );
    s.push( 18 );
    s.push( 19 );
    s.push( 20 );
    s.push( 21 );
    int i  = s.pop( );    cout << endl << "Item popped=" << i;
    i = s.pop( );         cout << endl << "Item popped=" << i;
    i = s.pop( );         cout << endl << "Item popped=" << i;
    i = s.pop( );         cout << endl << "Item popped=" << i;
}
```

31. Write program to implement a queue class with requried operations/ functions.

```
#include     <iostream.h>
#define     MAX     10

class queue
{

private:
    int arr[ MAX ];
    int front,
         rear;

public:
    queue( )
    {
        front = -1;
        rear  = -1;
    }

    void addq( )
    {
        int item;

        if( rear == MAX - 1 )
        {
            cout << endl << "Queue is full";
            return;
        }
        rear++;
        arr[ rear ] = item;
        if( front == -1 )
            front = 0;
    }

    int delq( )
    {
        int data;
        if( front == -1 )
        {
            cout << endl << "Queue is empty";
            return NULL;
        }

        data = arr[ front ];
        if( front == rear )
            front = rear = -1;
        else
            front++;

        return data;
    }
}    ;


void main( )
{
```

```
    queue  a;
    a.addq( 11 );
    a.addq( 12 );
    a.addq( 13 );
    a.addq( 14 );
    a.addq( 15 );
    a.addq( 16 );
    a.addq( 17 );
    a.addq( 18 );
    a.addq( 19 );
    a.addq( 20 );
    a.addq( 21 );

    int i  = a.delq( );
    cout << endl << "Item deleted=" << i;

    i = a.delq( );
    cout << endl << "Item deleted=" << i;

    i = a.delq( );
    cout << endl << "Item deleted=" << i;

}
```

32. Write a program to implement circular queue class with required operations/ functions.

```
#include     <iostream.h>
#define     MAX     10

class queue
{

private:
    int arr[ MAX ];
    int front,
         rear;
public:
    queue( )
    {
        front = -1;
        rear  = -1;
    }

    void addq( int  item )
    {
        if( ( rear == MAX - 1 && front == 0 )
             || ( rear + 1 == front ) )
        {
            cout << endl << "Queue is full";
            return;
        }

        if( rear == MAX - 1 )
            rear = 0;
        else
            rear = rear + 1;

        arr[ rear ] = item;

        if( front == -1 )
            front = 0;
    }

    int delq( )
    {
        int data;

        if( front == -1 )
        {
            cout << endl << "Queue is empty";
            return NULL;
        }
        else
        {
            data = arr[ front ];
            if( front == rear )
            {
                front = -1;
                rear  = -1;
            }
            else
            {
```

```
                if( front == MAX - 1 )
                    front = 0;
                else
                    front = front + 1;
            }
            return data;

        }
    }
}    ;

void main( )
{
    queue  a;
    a.addq( 11 );
    a.addq( 12 );
    a.addq( 13 );
    a.addq( 14 );
    a.addq( 15 );
    a.addq( 16 );
    a.addq( 17 );
    a.addq( 18 );
    a.addq( 19 );
    a.addq( 20 );
    a.addq( 21 );

    int i  = a.delq( );
    cout << endl << "Item deleted=" << i;

    i = a.delq( );
    cout << endl << "Item deleted=" << i;

    i = a.delq( );
    cout << endl << "Item deleted=" << i;

}
```

33. Write a program implementing linked list as a class. Also Perform some required operations like inserting, deleting nodes & display the contents of entire linked list.

```
#include     <iostream.h>

class linklist
{
    struct node
     {
        int    data;
        node  * link;
     }    * p;

public:
         linklist( );
    void append( int  num );
    void addatbeg( int  num );
    void addafter( int  c, int  num );
    void del( int  num );
    void display( );
    int  count( );
         ~linklist( );
}    ;

linklist::linklist( )
{
    p = NULL;
}

void linklist::append( int  num )
{
    node  * q,
          * t;
    if( p == NULL )
    {
        p       = new node;
        p->data = num;
        p->link = NULL;
    }
    else
    {
        q = p;
        while( q->link != NULL )
            q = q->link;
        t       = new node;
        t->data = num;
        t->link = NULL;
        q->link = t;
    }
}

void linklist::addatbeg( int  num )
{
    node  * q;
    q       = new node;
    q->data = num;
    q->link = p;
    p       = q;
```

```
}

void linklist::addafter( int  c, int  num )
{
    node  * q,
          * t;
    int    i;
    for( i = 0, q = p; i < c; i++ )
    {
        q = q->link;
        if( q = NULL )
        {
            cout << endl << "There are less than " << c << "element";
            return;
        }
    }
    t       = new node;
    t->data = num;
    t->link = q->link;
    q->link = t;
}

void linklist::del( int  num )
{
    node  * q,
          * r;
    q = p;
    if( q->data == num )
    {
        p = q->link;
        delete q;
        return;
    }
    r = q;
    while( q != NULL )
    {
        if( q->data == num )
        {
            r->link = q->link;
            delete q;
            return;
        }
        r = q;
        q = q->link;
    }
    cout << endl << "Element" << num << "not found";
}

void linklist::display( )
{
    node  * q;
    cout << endl;
    for( q = p; q->link != NULL; q = q->link )
    {
        cout << endl << q->data;
    }

int linklist::count( )
{
```

```
    node  * q;
    int    c  = 0;
    for( q = p; q != NULL; q = q->link )
        c++;
    return ( c );
}

linklist::~linklist( )
{
    node  * q;
    if( p == NULL )
        return;
    while( p != NULL )
    {
        q = p->link;
        delete p;
        p = q;
    }
}

void main( )
{
    linklist  ll;
    cout << endl << "No. of elements in linked list= " << ll.count( );
    ll.append( 11 );
    ll.append( 22 );
    ll.append( 33 );
    ll.append( 44 );
    ll.append( 55 );

    ll.addatbeg( 100 );
    ll.addatbeg( 200 );
    ll.addatbeg( 300 );

    ll.addafter( 3, 333 );
    ll.addafter( 6, 444 );

    ll.display( );
    cout << endl << "No. of element in linked list =" << ll.count( );

    ll.del( 300 );
    ll.del( 66 );
    ll.del( 0 );

    ll.display( );
    cout << endl << "No. of element in linked list =" << ll.count( );

}
```

34. Write a program implementing stack & its operations using dynamic memory allocation.

```
#include     <iostream.h>

struct node
{
    int    data;
    node  * link;
}    ;

class stack
{
private:
    node  * top;

public:
    stack( )
    {
        top = NULL;
    }

    void push( int  item )
    {
        node  * temp;
        temp = new node;
        if( temp = NULL )
            cout << endl << "Stack  is full";
        temp->data = item;
        temp->link = top;
        top        = temp;
    }

    int pop( )
    {
        if( top == NULL )
        {
            cout << endl << "Stack is empty";
            return NULL;
        }
        node  * temp;
        int    item;
        temp = top;
        item = temp->data;
        top  = top->link;
        delete temp;
        return item;
    }

    ~stack( )
    {
        if( top == NULL )
            return;
        node  * temp;
        while( top != NULL )
        {
            temp = top;
            top  = top->link;
            delete temp;
```

```
        }
    }
}    ;

void main( )
{
    stack  s;
    s.push( 11 );
    s.push( 12 );
    s.push( 13 );
    s.push( 14 );
    s.push( 15 );
    s.push( 16 );

    int i  = s.pop( );
    cout << endl << "Item popped=" << i;

    i = s.pop( );
    cout << endl << "Item popped=" << i;

    i = s.pop( );
    cout << endl << "Item popped=" << i;
}
```

35. Write a program implementing Queue stack & its operations using dynamic memory allocation.

```
#include     <iostream.h>

struct node
{
    int    data;
    node  * link;
}    ;

class queue
{

private:
    node  * front,
          * rear;
public:
    queue( )
    {
        front = rear = NULL;
    }

    void addq( int  item )
    {
        node  * temp;
        temp = new node;
        if( temp == NULL )
            cout << endl << "Queue is full";
        temp->data = item;
        temp->link = NULL;
        if( front == NULL )
        {
            rear = front = temp;
            return;
        }
        rear->link = temp;
        rear       = rear->link;
    }

    int delq( )
    {
        if( front == NULL )
        {
            cout << endl << "queue is empty";
            return NULL;
        }
        node  * temp;
        int    item;
        item  = front->data;
        temp  = front;
        front = front->link;
        delete temp;
        return item;
    }


    ~queue( )
    {
```

```
        if( front == NULL )
            return;
        node  * temp;
        while( front != NULL )
        {
            temp  = front;
            front = front->link;
            delete temp;
        }

    }
}    ;

void main( )
{
    queue  a;
    a.addq( 11 );
    a.addq( 12 );
    a.addq( 13 );
    a.addq( 14 );
    a.addq( 15 );
    a.addq( 16 );
    a.addq( 17 );

    int i  = a.delq( );
    cout << endl << "Item extracted=" << i;

    i = a.delq( );
    cout << endl << "Item extracted=" << i;

    i = a.delq( );
    cout << endl << "Item extracted=" << i;
}
```

36. Write a program to implement Binary search tree using class and traverse the tree using any traversal scheme. In addition to it the class must have capability to copy the contents from one tree to another and compare the contents of two binary trees.

```
#include     <iostream.h>
#define     TRUE      1
#define     FALSE     0

class tree
{
private:
    struct node
     {
        node  * l;
        int    data;
        node  * r;
     }    * p;
public:
            tree( );
    void    search( int  n, int & found,
                    node * parent );
    void    insert( int  n );
    void    traverse( );
    int     in( node * q );
    void    pre( node * q );
    void    post( node * q );
    int     operator ==( tree  t );
    int     compare( node * pp, node * qq );
    void    operator =( tree  t );
    node  * copy( node * q );
}    ;

tree::tree( )
{
    p = NULL;
}

void tree::search( int  n, int & found, node *& parent )
{
    node  * q;
    found  = FALSE;
    parent = TRUE;
    if( p == NULL )
        return;
    q = p;
    while( q != NULL )
    {
        if( q->data == n )
        {
            found = TRUE;
            return;
        }
        if( q->data > n )
        {
            parent = q;
            q      = q->l;
        }
        else
```

```
        {
            parent = q;
            q      = q->r;
        }
    }
}

void tree::insert( int  n )
{
    int    found;
    node  * t,
          * parent;
    search( n, found, parent );
    if( found == TRUE )
        cout << endl << "Such a node already exist";
    else
    {
        t       = new node;
        t->data = n;
        t->l    = NULL;
        t->r    = NULL;
        if( parent == NULL )
            p = t;
        else
            parent->data > n?parent->l:parent->r = t;
    }
}

void tree::traverse( )
{
    int choice;
    cout << endl << "q.Inorder" << endl << "2. Preorder" << endl
     << "3. Postorder" << endl << "4. Your choice ";
    cin >> choice;
    switch( choice )
    {
        case 1:
            in( p );
            break;
        case 2:
            pre( p );
            break;
        case 3:
            post( p );
            break;
    }
}

void tree::in( node * q )
{
    if( q != NULL )
    {
        in( q->l );
        cout << "\t" << q->data;
        in( q->r );
    }
}
void tree::pre( node * q )
{
```

```
    if( q != NULL )
    {
        cout << "\t" << q->data;
        pre( q->l );
        pre( q->r );
    }
}

void tree::post( node * q )
{
    if( q != NULL )
    {
        post( q->l );
        post( q->r );
        cout << "\t" << q->data;
    }
}

int tree::operator ==( tree  t )
{
    int flag;
    flag = compare( p, t.p );
    return ( flag );
}

int tree::compare( node * pp, node * qq )
{
    static  int flag;
    if( ( pp == NULL ) && ( q != NULL ) )
    {
        if( ( pp != NULL ) && ( qq != NULL ) )
        {
            if( pp->data != qq->data )
                flag = FALSE;
            else
            {
                compare( pp->l, qq->l );
                compare( qq->r, qq->r );
            }
        }
    }
    return ( flag );
}

void tree::operator =( tree  t )
{
    p = copy( t.p );
}

tree::node    * tree::copy( node * q )
{
    if( q != NULL )
    {
        t       = new node;
        t->data = q->data;
        t->l    = copy( q->l );
        t->r    = copy( q->r );
        return ( t );
    }
```

```
    else
        return ( NULL );
}

void main( )
{
    tree  tt,
          ss;
    int   i,
          num;

    for( i = 0; i <= 6; i++ )
    {
        cout << endl << "Enter the data for the node to be inserted";
        cin >> num;
        tt.insert( num );
    }

    tt.traverse( );
    ss = tt;
    ss.traverse( );

    if( ss == tt )
        cout << endl << "Trees are equal";
    else
        cout << endl << "Trees are not equal";
}
```

37. Write a program to implement the exception handling with multiple *catch* statements.

```
#include<iostream.h>

void test(int x)
{
      try
      {
            if(x==1)
                  throw x;
            else
                  if(x==0)
                        throw 'x';
                  else
                        if(x==-1)
                              throw 1.0;
                        cout<<"End of try-black\n";
      }

      catch(char c)
      {
            cout<<"Caught a Character\n";
      }

      catch(int c)
      {
            cout<<"Caught an Integer\n";
      }

      catch(double c)
      {
            cout<<"Caught a Double\n";
      }

      cout<<"End of try-catch system\n";
}

int main()
{
      cout<<"Testing Multiple Catches\n";
      cout<<"x==1\n";
      test(1);
      cout<<"x==0\n";
      test(0);
      cout<<"x==2\n";
      test(2);

      return 0;
}
```

38. Write a program to implement the exception handling with rethrowing in exception.

```
#include<iostream.h>

void divide(double x, double y)
{
      cout<<"Inside Function\n";
      try
      {
            if(y==0.0)
                  throw y;
            else
                  cout<<"Division ="<<x/y<<"\n";
      }

      catch(double)
      {
            cout<<"Caught double inside function\n";
            throw;
      }

      cout<<"End of Function\n";
}

int main()
{
      cout<<"Inside Main\n";
      try
      {
            divide(10.5,2.0);
            divide(20.0,0.0);
      }

      catch(double)
      {
            cout<<"Caught double inside main\n";
      }

      cout<<"End of Main\n";

      return 0;
}
```

39. Write a program to implement the exception handling with the functionality of testing the *throw* restrictions.

```
#include<iostream.h>

void test(int x) throw(int, double)
{
      if(x==0)
            throw 'x';
      else
            if(x == 1)
                  throw x;
            else
                  if(x == -1)
                        throw 1.0;
                  cout<<"End of Function Block\n";
}

int main()
{
      try
      {
            cout<<"Testting Throw Restrictions\n";
            cout<<"x == 0\n";
            test(0);
            cout<<"x == 1\n";
            test(1);
            cout<<"x == -1\n";
            test(-1);
            cout<<"x == 2\n";
            test(2);
      }
      catch(char c)
      {
            cout<<"Caught a Character\n";
      }

      catch(int m)
      {
            cout<<"Caught an Integer\n";
      }

      catch(double d)
      {
            cout<<"Caught a Double\n";
      }

      cout<<"End of Try-catch system\n";
      return 0;
}
```

40. Write a function template that will sort an array of implicit types like int,float,char etc. it can also sort user-defined objects like strings & date. The necessary classes contains overloading of operators.

```
#include     <iostream.h>
#include     <string.h>

class mystring
{
private:
     enum
     {
        sz = 100                                          //    < >
     }    ;
    char str[ sz ];

public:
    mystring( char * s = "" )
    {
        strcpy( str, s );
    }

    int operator <( mystring  ss )
    {
        if( strcmp( str, ss.str ) <= 0 )
            return 1;
        else
            return 0;
    }

    int operator <=( mystring  ss )
    {
        if( strcmp( str, ss.str ) <= 0 )
            return 1;
        else
            return 0;
    }

    int operator >( mystring  ss )
    {
        if( strcmp( str, ss.str ) > 0 )
            return 1;
        else
            return 0;
    }

    friend  ostream  & operator <<( ostream & o,mystring & dd );

}    ;

ostream  operator <<( ostream & o, mystring & ss )
{
    o << ss.str;
    return o;
}


class date
{
```

```
private:
    int day,
         mth,
         yr;
public:
    date( int  d = 0, int  m = 0, int  y = 0 )
    {
        day = d;
        mth = m;
        yr  = y;
    }

    int operator <( date  dt )
    {
        if( yr < dt.yr )
            return 1;
        if( yr == dt.yr && mth < dt.mth )
            return 1;
        if( yr == dt.yr && mth == dt.mth && day = dt.day )
            return 1;
        return 0;
    }

    class date
     {
    private:
        int day, mth, yr;
    public:
        date( int  d = 0, int  m = 0, int  y = 0 )
        {
            day = d;
            mth = m;
            yr  = y;
        }

        int operator <( date  dt )
        {
            if( yr < dt.yr )
                return 1;
            if( yr == dt.yr && mth < dt.mth )
                return 1;
            if( yr == dt.yr && mth == dt.mth
                 && day < dt.day )
                return 1;
            return 0;
        }
        int operator <=( date  dt )
        {
            if( yr <= dt.yr )
                return 1;
            if( yr == dt.yr && mth <= dt.mth )
                return 1;
            if( yr == dt.yr && mth == dt.mth
                 && day <= dt.yr )
                return 1;
            return 0;
        }
        int operator >( date  dt )
        {
```

```
        if( yr > dt.yr )
            return 1;
        if( yr == dt.yr && mth > dt.mth )
            return 1;
        if( yr == dt.yr && mth == dt.mth
             && day > dt.day )
            return 1;
        return 0;
    }

    friend  ostream  & operator <<( ostream & o, date & dd );
 }    ;

ostream  & operator <<( ostream & o, date & dd )
{
    o << dd.day << "\t" << dd.mth << "\t" << dd.yr;
    return 0;
}

template<class T>  void quick( T * n, int  low, int  high )
{
    int pos;
    if( low < high )
    {
        pos = split( n, low, high );
        quick( n, low, pos - 1 );
        quick( n, pos + 1, high );
    }
}

template<class T>  int split( T * n, int  low, int  high )
{
    int pos,
         left,
         right;
    T   item, t;

    item  = n[ low ];
    left  = low;
    right = high;

    while( left < right )
    {
        while( n[ right ] > item )
            right = right - 1;

        while( ( left < right )
                && ( n[ left ] <= item ) )
            left = left + 1;

        if( left < right )
        {
            t          = n[ left ];
            n[ left ]  = n[ right ];
            n[ right ] = t;
        }
    }
    pos      = right;
    t        = n[ low ];
```

```
    n[ low ] = n[ pos ];
    n[ pos ] = t;
    return pos;
}


void main( )
{
    float num[]={5.4f,3.23f,2.15f,1.09f,34.66f,23.3452f};
    int arr[]={-12,23,14,0,245,78,66,-9};

    date dtarr[]={date(17,11,62),date(23,12,65),date(12,12,78)
                    ,date(23,1,69)};

    mystring strarr[]={mystring("Kamal"),mystring("Anuj"),
                            mystring("Sachin"),mystring("Anil")};

    int       i;
    cout << endl << endl;

    quick( num, 0, 5 );
    for( i = 0; i <= 5; i++ )
        cout << num[ i ] << endl;

    cout << endl << endl;
    quick( arr, 0, 7 );
    for( i = 0; i <= 7; i++ )
        cout << arr[ i ] << endl;

    cout << endl << endl;
    quick( dtarr, 0, 3 );
    for( i = 0; i <= 3; i++ )
        cout << dtarr[ i ] << endl;

    cout << endl << endl;
    quick( strarr, 0, 3 );
    for( i = 0; i <= 3; i++ )
        cout << strarr[ i ] << endl;

}
```

41. Write a program implementing stack and it's operations using template class.

```
#include     <iostream.h>

const  int MAX  = 10;

template<class T>class stack
{
private:
    T   stk[ MAX ];
    int top;

public:
    stack( )
    {
        top = -1;
    }
    void push( T  data )
    {
        if( top == MAX - 1 )
            cout << endl << "Stack is full";
        else
        {
            top++;
            stk[ top ] = data;
        }
    }

    T  pop( )
    {
        if( top == -1 )
        {
            cout << endl << "Stack is empty";
            return NULL;
        }
        else
        {
            T  data  = stk[ top ];
            top--;
            return data;
        }
    }
}    ;

class complex
{
private:
    float real,
          imag;

public:
    complex( float  r = 0.0, float  i = 0.0 )
    {
        real = r;
        imag = i;
    }
    friend  ostream  & operator <<( ostream & o,
                                    complex & c );
```

```
}    ;

ostream  & operator <<( ostream & o, complex & c )
{
    o << c.real << "\t" << c.imag;
    return o;
}

void main( )
{
    stack< int >    s1;
    s1.push( 10 );
    s1.push( 20 );
    s1.push( 30 );

    cout << endl << s1.pop( );
    cout << endl << s1.pop( );
    cout << endl << s1.pop( );

    stack< float >    s2;

    s2.push( 3.14 );
    s2.push( 6.28 );
    s2.push( 8.98 );

    cout << endl << s2.pop( );
    cout << endl << s2.pop( );
    cout << endl << s2.pop( );

    complex             c1 ( 1.5, 2.5 ),
                         c2 ( 3.5, 4.5 ),
                         c3 ( -1.5, -0.6 );

    stack< complex >    s3;

    s3.push( c1 );
    s3.push( c2 );
    s3.push( c3 );

    cout << endl << s3.pop( );
    cout << endl << s3.pop( );
    cout << endl << s3.pop( );

}
```

42. Write a program implementing linked list & some required operations on it using class template.

```
#include     <string.h>
#include     <iostream.h>

class emp
{
private:
    char  name[ 20 ];
    int   age;
    float sal;

public:
    emp( char * n = "", int  a = 0, float  s = 0.0 )
    {
        strcpy( name, n );
        age = a;
        sal = s;
    }

    friend  ostream  & operator <<( ostream & s, emp & e );

}    ;
ostream  operator <<( ostream & s, emp & e )
{
    cout << e.name << "\t" << e.age << "\t" << e.sal;
    return s;
}

template<class T>class linklist
{
private:
    struct node
     {
        T      data;
        node  * link;
     }    * p;

public:
    linklist( );
    ~linklist( );
    void append( T );
    void addatbeg( T );
    void addafter( int, T );
    void del( int );
    void display( );
    int count( );
}    ;

template<class T>  linklist< T >::linklist( )
{
    p = NULL;
}


template<class T>  linklist< T >::~linklist( )
{
    node  * t;
```

```
    while( p != NULL )
    {
        t = p;
        p = p->link;
        delete t;
    }
}

template<class T>  void linklist< T >::append( T  num )
{
    node  * q,
          * t;
    if( p == NULL )
    {
        p       = new node;
        p->data = num;
        p->link = NULL;
    }
    else
    {
        q = p;
        while( q->link != NULL )
            q = q->link;
        t       = new node;
        t->data = num;
        t->link = NULL;
        q->link = t;
    }
}

template<class T>  void linklist< T >::addatbeg( T  num )
{
    node  * q;
    q       = new node;
    q->data = num;
    q->link = p;
    p       = q;
}

template<class T>  void linklist< T >::addafter( int  c,
                                                 T  num )
{
    node  * q,
          * t;
    int    i;

    for( i = q, q = p; i <= c; i++ )
    {
        q = q->link;
        if( q == NULL )
        {
            cout << endl << "There are less than" << c << "element";
            return;
        }
    }

    t       = new node;
    t->data = num;
    t->link = q->link;
```

```
    q->link = t;
}

template<class T>  void linklist< T >::del( int  n )
{
    node  * q,
          * r;
    int    i  = 1;
    q = p;
    if( n == 1 )
    {
        p = q->link;
        delete q;
        return;
    }
    r = q;
    while( q != NULL )
    {
        if( i == n )
        {
            r->link = q->link;
            delete q;
            return;
        }
        r = q;
        q = q->link;
        i++;
    }
    cout << endl << "Element" << n << "not found";
}

template<class T>  void linklist< T >::display( )
{
    node  * q;
    cout << endl;
    for( q = p; q != NULL; q = q->link )
        cout << q->data << endl;
}

template<class T>  int linklist< T >::count( )
{
    node  * q;
    int    c  = 0;
    for( q = p; q != NULL; q = q->link )
        c++;
    return ( c );
}

void main( )
{
    linklist< int >    l1;
    cout << endl << "No. of elements in linked list = " << l1.count( );

    l1.append( 11 );
    l1.append( 22 );
    l1.append( 33 );
    l1.append( 44 );
    l1.append( 55 );
    l1.append( 66 );
```

```
    l1.addatbeg( 100 );
    l1.addatbeg( 200 );

    l1.addafter( 3, 333 );
    l1.addafter( 4, 444 );

    l1.display( );

    cout << endl << "No. of elements in linked list=" << l1.count( );

    l1.del( 200 );
    l1.del( 66 );
    l1.del( 0 );
    l1.del( 333 );

    l1.display( );

    cout << endl << "no. of elements in linked list = " << l1.count( );
    linklist< emp >    l2;

    cout << endl << "No. of elements in linked list = " << l2.count( );

    emp  e1 ( "Sanjay", 23, 1100.00 );
    emp  e2 ( "Rahul", 33, 3500.00 );
    emp  e3 ( "Rakesh", 24, 2400.00 );
    emp  e4 ( "Sanket", 25, 2500.00 );
    emp  e5 ( "Sandeep", 26, 2600.00 );

    l2.append( e1 );
    l2.append( e2 );
    l2.append( e3 );
    l2.append( e4 );
    l2.append( e5 );

    l2.display( );

    l2.del( 3 );
    l2.display( );

    cout << endl << "No. of elements in linked list = " << l2.count( );
    l2.addatbeg( e5 );
    l2.display( );

    l2.addafter( 3, e1 );
    l2.display( );

    cout << endl << "No. of elements in linked list = " <<  l2.count( );
}
```

43. Write a program using mouse service routine (0x33 interrupt). The program should track all mouse activities.

```
//mouse.cpp
#include     <iostream.h>
class mouse
{
private:
    union  REGS  i,
                  o;
public:
    mosue( )
    {
        initmouse( );
        showmouseptr( );
    }

    void initmouse( )
    {
        i.x.ax = 0;
        int86( 0x33, &i, &o );
    }

    void showmouseptr( )
    {
        i.x.ax = 1;
        int86( 0x33, *i, &o );
    }

    void hidemouseptr( )
    {
        i.x.ax = 2;
        int86( 0x33, &i, &o );
    }

    void getmousepos( int & button, int & x, int & y )
    {
        i.x.ax = 3;
        int86( 0x33, &i, &o );
        button = o.x.bx;
        x      = o.x.cx;
        y      = o.x.dx;
    }

    void restrictmouseptr( int  x1, int  y1, int  x2, int  y2 )
    {
        i.x.ax = 7;
        i.x.cx = x1;
        i.x.dx = x2;
        int86( 0x33, &i, &o );
        i.x.ax = 8;
        i.x.cx = y1;
        i.x.dx = y2;
        int86( 0x33, &i, &o );
    }
}    ;

//Virtual.cpp
```

```
#include     <iostream.h>
#include     <stdio.h>
#include     <string.h>
#include     <stdlib.h>
#include     <graphics.h>
#include     <conio.h>
#include     <dos.h>
#include     "mouse.cpp"
#include     <fstream.h>

class shapes
{
public:
    virtual  void draw( )
    {
    }
    virtual  void save( ofstream & ft )
    {
    }
    virtual  void open( ifstream & fs )
    {
    }
}    ;

class myline:public shapes
{
private:
    int sx,
         sy,
         ex,
         ey,
         color;
public:
    myline( )
    {
    }

    myline( int  x1, int  y1, int  x2, int  y2,int  clr )
    {
        sx    = x1;
        sy    = y1;
        ex    = x2;
        ey    = y2;
        color = clr;
    }
    void draw( )
    {
        setcolor( color );
        moveto( sx, sy );
        lineto( ex, ey );
    }

    void save( ofstream & ft )
    {
        ft << "R" << "\n";
        ft <<sx<<""<<sy<<""<<ex<<""<< ey << "" << color << "\n";
    }

    void open( ifstream & fs )
```

```
    {
        fs >> sx >> sy >> ex >> ey >> color;
    }
}    ;

class myrectangle:public shapes
{
private:
    int sx,
         sy,
         ex,
         ey,
         color;
public:
    myrectangle( )
    {
    }

    myrectangle( int  x1, int  y1, int  x2, int  y2,int  clr )
    {
        sx    = x1;
        sy    = y1;
        ex    = x2;
        ey    = y2;
        color = clr;
    }

    void draw( )
    {
        setcolor( color );
        rectangle( sx, sy, ex, ey );
    }

    void save( ofstream & ft )
    {
        ft << "R" << "\n";
        ft <<sx<<""<<sy<<""<< ex << "" << ey << "" << color << endl;
    }

    void open( ifstream & fs )
    {
        fs >> sx >> sy >> ex >> ey >> color;
    }
}    ;

class mycircle:public shapes
{
private:
    int sx,
         radius,
         color;

public:
    mycircle( )
    {
    }

    mycircle( int  x1, int  y1, int  r, int  clr )
    {
```

```
        sx     = x1;
        sy     = y1;
        radius = r;
        color  = clr;
    }

    void draw( )
    {
        setcolor( color );
        circle( sx, sy, radius );
    }

    void save( ofstream & ft )
    {
        ft << "C" << "\n";
        ft << sx << "" << sy << "" << radius << "" << color << endl;
    }

    void open( ifstream & fs )
    {
        fs >> sx >> sy >> radius >> color;
    }
}    ;

struct node
{
    void  * obj;
    node  * link;
}    ;

class objarray
{
private:
    node  * head;
public:

    objarray( )
    {
        head = NULL;
    }

    void add( void * o )
    {
        node  * temp  = new node;
        temp->obj  = o;
        temp->link = NULL;
        if( head == NULL )
            head = temp;
        else
        {
            node  * q;
            q = head;
            while( q->link != NULL )
                q = q->link;
            q->link = temp;
        }
    }

    void * getobj( int  i )
```

```
    {
        node  * q;
        q = head;
        int n;
        for( n = 1; n < i; n++ )
        {
            q = q->link;
        }
        return ( q->obj );
    }

    int getcount( )
    {
        int    n  = 0;
        node  * q;
        q = head;
        while( q != NULL )
        {
            q = q->link;
            n++;
        }
        return n;
    }

    ~objarray( )
    {
        node  * q;
        q = head;
        while( q != NULL )
        {
            head = head->link;
            delete q;
            q = head;
        }
    }
}    ;

void mainscreen( )
{
    clearddevice( );
    rectangle( 0, 0, 639, 479 );
    line( 0, 30, 640, 30 );
    char *names[]={"Clear","Open","Save","Line","Rect","Circ",
                        "Exit"};
    int     x, i;
    for( x = 5, i = 0; x <= 7 * 90; x += 90, i++ )
    {
        setcolor( WHITE );
        rectangle( x, 5, x + 70, 25 );
        floodfill( x + 1, 6, WHITE );
        settextstyle( 1, 0, 3 );
        setcolor( BLACK );
        outtextxy( x + 10, 0, names[ i ] );
    }
}


void main( )
{
```

```
ifstream  fs;
ofstream  ft;
int       gd  = DETECT, gm;

initgraph( &gd, &gm, "c:\\tc\\bgi" );
mainscreen( );
setviewport( 1, 31, 638, 478, 1 );
mouse  m;
int    button,
        x,
        y,
        flag  = 0;
int    strptx,
        strpty,
        endptx,
        endpty;
objarray arrl
while( 1 )
{
    button = 0;
    m.getmousepos( button, x, y );
    if( ( button & q ) == 1 ) &&(flag==0))
    {
        for( t = 5, i = 0; t <= 7 * 90; t += 90, i++ )
        {
            if( x >= t && x <= t + 70 && y >= 5&& y <= 25 )
            {
                index = i;
                flag  = 1;
                break;
            }
        }
    }
    int cirnum  = random( 16 );
    int sx  = random( 638 );
    int sy  = random( 478 );
    int ex  = random( 638 );
    int ey  = random( 478 );
    int r  = random( 200 );

    switch( index )
    {
        case 0:
            m.getmousepos( button, x, y );
            if( ( ( button & 1 ) == 0 ) && ( flag == 1 ) )
            {
                clearviewport( );
                flag = 0;
            }
            break;

        case 1:
            m.getmousepos( button, x, y );
            if( ( ( button & 1 ) == 0 ) && ( flag == 1 ) )
            {
                fs.open( "output.txt", ios::in );
                shapres * ptr;
                char a[ 2 ];
                while( fs )
```

```
        {
            fs >> a;
            if( strcmp( a, "L" ) == 0 )
            {
                myline  * l  = new myline(  );
                l->open( fs );
                arr.add( l );
            }
            if( strcmp( a, "R" ) == 0 )
            {
                myrectangle  * r  = new myrectangle(  );
                c->open( fs );
                arr.add( c );
            }
        }
        fs.close( );
        int count  = arr.getcount( );
        for( int  i  = 1; i <= count; i++ )
        {
            ptr = ( shapres * )arr.getobj( i );
            ptr->draw( );
        }
        flag = 0;
    }
    break;

case 2:
    m.getmousepos( button, x, y );
    if( ( ( button & 1 ) == 0 ) && ( flag == 1 ) )
    {
        ft.open( "output.txt", ios::out );
        int       count  = arr.getcount( );
        shapres  * ptr;
        for( i = 1; i <= count; i++ )
        {
            ptr = ( shapres * )arr.getobj( i );
            ptr->save( ft );
        }
        ft.close( );
        flag = 0;
    }
    break;

case 3:
    m.getmousepos( button, x, y );
    if( ( ( button & 1 ) == 0 )&& ( flag == 1 ) )
    {
        setcolor( clrnum );
        moveto( sx, sy );
        lineto( ex, ey );
        myline  *  l  = new myline
                              ( sx, sy,
                                ex, ey,
                                clrnum
                               );
        {
            if( l == NULL )
                exit( 1 );
            arr.add( l );
```

```
                    flag = 0;
                }
                break;

        case 4:
                m.getmousepos( button, x, y );
                if( ( ( button & 1 ) == 0 ) && ( flag == 1 ) )
                {
                    setcolor( clrnum );
                    rectangle( sx, sy, ex, ey );
                    myrectangle  *  r  = new myrectangle( sx, sy,
                                                    ex, ey, clrnum );
                    if( r == NULL )
                        exit( 1 );
                    arr.add( r );
                    flag = 0;
                }
                break;

        case 5:
                m.getmousepos( button, x, y );
                if( ( ( button & 1 ) == 0 ) && ( flag == 1 ) )
                {
                    setcolor( clrnum );
                    circle( sx, sy, r );
                    mycircle *c  = new mycircle(sx,sy,r,clrnum );
                    if( c == NULL )
                        exit( 1 );
                    arr.add( c );
                    flag = 0;
                }
                break;
            }
            if( index == 6 )
                break;
    }

    closegraph( );
    restorecrtmode( );
}
```