COOCHBEHAR GOVERNMENT ENGINEERING COLLEGE

PROGRAMMING PRACTICES USING C++
(CS 593)
(2018-2019)

<u>SUBMITTED BY-</u>

SOUMEN MAHATA

B.TECH (5TH SEMESTER)

ROLL NO. 34900117004

REG NO. 173490120013

S.N.	PROGRAMMES
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.
2.	Write a Program using class to process Shopping List for a Departmental Store. The list includes 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 an Order.
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..).
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.
5.	Write a Program to swap private data members of classes named as class_1, class_2 using friend function.
6.	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.

7.	Write a Program using copy constructor to copy data of an object to another object.
8.	Write a Program to allocate memory dynamically for an object of a given class using class's constructor.
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.
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.
11.	Write a Program to overload operators like *, <<, >> using friend function. The following overloaded operators should work for a class vector.
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.
13.	Write a program to overload new/delete operators in a class.
14.	Write a program in C++ to highlight the difference between overloaded assignment operator and copy constructor.
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.
16.	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.
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.

18.	Write a Program to illustrate the use of pointers to objects which are related by inheritance.
19.	Write a program illustrating the use of virtual functions in class.
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.

21.	Write a program to show conversion from string to int and vice-versa.
22.	Write a program showing data conversion between objects of different classes.
23.	
24.	Write a program to implement I/O operations on characters. I/O operations includes inputting a string, calculating length of the string, Storing the string in a file, fetching the stored characters from it, etc.
25.	Write a program to copy the contents of one file to another.
26.	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).
27.	Write a program to maintain an elementary database of employees using files.
28	Write a program for reading and writing data to and from the file using command line arguments.
29.	Write a program showing implementation of stack class having the functionality of push, pop operations.
30.	Write program to implement a queue class with required operations/ functions.
31.	Write a program to implement circular queue class with required operations/ functions.
32.	

33.	Write a program implementing stack & its operations using dynamic memory allocation.
34.	Write a program implementing Queue stack & its operations using dynamic memory allocation.
35.	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.
36.	Write a program to implement the exception handling with multiple catch statements.
37.	Write a program to implement the exception handling with throwing in exception.
38.	Write a program to implement the exception handling with the functionality of testing the throw restrictions.
39.	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 contain overloading of operators.
40.	Write a program implementing stack and its operations using template class.
41.	Write a program implementing linked list & some required operations on it using class template
42.	Write a program using mouse service routine (0x33 interrupt). The program should track all mouse activities.

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.**

Program:

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

Output:

```
Count:3

Object number:1

Object number:2

Object number:3
```

2. **Write a Program using class to process Shopping List for a Departmental Store. The list includes 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 an Order.**

Program:

```
  #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]; 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;
}
```

Output:

```
  You can do the following? Enter appropiat number
 1 : Add an item
 2: Display total value
 3: Delete an item
 4: Display all item
 5: Quit
```

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>
using namespace std;        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; }
```

Output: 20

5. **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(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;
}
```

Output
:

```
Values before Exchange
100
200
Values after exchange
200
100
```

6. **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> using
namespace std;  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;
}
```

Output
:

```
A  = 3.5 + j5.65
```

B = 2.75 + j1.2
C = 5.85 + j6.5

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

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

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;
}
```

Output
:

```
Id of A: 100
Id of B: 100
Id of C: 100
```

Id of D: 100

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

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

Output
:

```
Joseph
Louis
Lagrange
Joseph Louis
JosephLouis Lagrange
```

9. **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; 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;
}
```

Output:

```
Enter the size of matrix 2
2
Enter matrix element row by row 1
2
3
4
8651088
```

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

```
#include<string.h> struct
complex
{ float real;
float imag; };
```

```
complex operator + (complex a,complex b); complex
operator - (complex a,complex b); complex operator *
(complex a,complex b); complex operator / (complex
a,complex b);
 void main() {
complex a,b,c;
int ch;
void menu(void);clrscr(); cout<<"Enter the first
complex no:"; cin>>a.real>>a.imag;
cout<<"Enter the second complex no:";
cin>>b.real>>b.imag; menu();
while ((ch = getchar()) != 'q')
{ switch(ch) {
case 'a':c =a + b; cout<<"Addition
of 2 no’s";
cout<<c.real<<"+i"<<c.imag;
break; case 's':c=a-b;
cout<<"Substraction of 2 no’s";
cout<<c.real<<"i"<<c.imag; break;
case 'm':c=a*b;
cout<<"Multiplication of 2 no’s";
cout<<c.real<<"i"<<c.imag; break;
                              case 'd':c=a/b;
cout<<"Division of 2 no’s";
cout<<c.real<<"i"<<c.imag; break;
                      }
              }
       }
        void menu()
{
cout<<"complex no: operators";
cout<<"a->addition"; cout<<"s-
>substraction"; cout<<"m-
>multiplication"; cout<<"d->division";
cout<<"q->quit"; cout<<"options
please";
}
complex operator -(struct complex a, struct complex b)
{ complex c;
c.real=a.real-b.real;
c.imag=a.imag-b.imag;
return(c); }
complex operator *(struct complex a, struct complex b)
{ complex c;
c.real=((a.real*b.real)-(a.imag*b.imag));
c.imag=((a.real*b.imag)+(a.imag*b.real)); return(c);
}
complex operator +(struct complex a,struct complex b)
{ complex c;
c.real=a.real+b.real;
c.imag=a.imag+b.imag;
return(c); }
complex operator /(struct complex a, struct complex b)
{ complex c;
float temp;
temp=((b.real*b.real)+(b.imag*b.imag));
c.real=((a.real*b.real)+(a.imag*b.imag))/temp;
                    return(c);
}
```

Output

```
Enter the first complex no: 1,1
Enter the second complex no: 2,2

Addition of 2 no's : 3+I3
```

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

```
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;
}
```

Output:

12. **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;

class mat
{
private:
                int s[10][10];
int u,v;        public:
void show();                    mat
operator +(mat);
mat operator *(mat);
                void read();
};
        mat mat::operator+(mat uu2)
    {
```

```
               mat t;
t.u=u;
               t.v=v;
cout<<t.u;
cout<<t.v;
for(int i=0;i<t.u;i++)
for(int j=0;j<t.v;j++)
                         t.s[i][i]=s[i][i]+uu2.s[i][i];
return t;
      }
          mat mat::operator*(mat uu2)
     {
mat t;
t.u=u;
               t.v=uu2.v;
for(int i=0;i<t.u;i++)
                    for(int j=0;j<t.v;j++)
                        {
                          t.s[i][i]=0;
for(int k=0;k<v;k++)

t.s[i][j]+=s[i][k]*uu2.s[k][j];                         }
return t;
       }
           void mat::read()
       {
              cout<<"Enter Size of Matrix like 3 x
3:\n";                cin>>u>>v;
              cout<<"Enter the Elements of Matrix
:\n";
              for(int i=0;i<u;i++)
for(int j=0;j<v;j++)
cin>>s[i][j];
     }
```

```
            void mat::show()
    {              for(int
i=0;i<u;i++)
                {
for(int j=0;j<v;j++)
                    {
                        cout<<s[i][j]<<"\t";

}
cout<<"\n";
                }
    }
            void main()
    {
                mat obj1 ,obj2,obj3;
clrscr();
                cout<<"Enter First Matrix\n";
obj1.read();
                cout<<"Enter Second Matrix\n";
                obj2.read();
obj3=obj1 +obj2;
                cout<<"Result After Addition of two Matrix\n";
obj3.show();              obj3=obj1 *obj2;
cout<<"Result After Multiplication of two Matrix\n";
obj3.show();              getch();
    }
```

Output:

```
Enter first matrix
Enter size of matrix like 3 X 3:
2
2
Enter Elements of Matrix :
1
2
```

```
3
4
Enter Second Matrix:
Enter size of Matrix like
3x3: 2
2
Enter Elements of Matrix
1
2
3
4
Result After Addition of two matrix
2    4
6     8
Result After Multiplication of two Matrix
```

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

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

using   namespace   std;      class
student
{
   string         name;
int   age;       public:
student()       {
       cout<< "Constructor is called\n" ;
   }
   student(string name, int age)
   {
       this->name = name;              this->age =
age;
   }
   void display()
   {
       cout<< "Name:" << name << endl;          cout<< "Age:"
<< age << endl;
   }
   void * operator new(size_t size)
   {
       cout<< "Overloading new operator with size: "
<< size << endl;
       void * p = ::new student();
       //void * p = malloc(size); will also work fine

       return p;      }

   void operator delete(void * p)
   {
       cout<< "Overloading delete operator " << endl;           free(p);
}
};
   int main()  {       student * p = new student("Yash",
24);

   p->display();      delete p;
}
```

Output

```
Overloading new operator with size: 16
Constructor is called
Name:Yash
Age:24
```

Overloading delete operator

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

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

using namespace std;

class Test
{  public:
Test() {}
   Test(const Test &t)
   {
      cout<<"Copy constructor called "<<endl;
   }
   Test& operator = (const Test &t)
   {
      cout<<"Assignment operator called "<<endl;
   }
};

int main()
{
  Test t1, t2;    t2 =
t1;    Test t3 = t1;
getchar();    return
0;  }
```

Output

```
Assignment operator called
Copy constructor called
```

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>
using namespace std;

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";
} }; int main()
{
gamma g(5, 10.75, 20, 30); g.show_x();
g.show_y();
g.show_mn();
return 0;
}
```

Output:

```
Beta initialized
Alpha initialized
Gamma initialized
X = 5 Y =
 10.75  m = 20
n = 30
```

16. **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>
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;

student_1.get_number (1234);
```

```
student_1.get_marks (27.5, 33.0);

student_1.get_score (6.0);

student_1.display ();

return 0;
}
```

```
Output
Roll No : 1234
Marks Obtained
Part1 = 27.6
Part2 = 33
Sports wt : 6
Total Score: 66.5
```

17. **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>
#include<conio.h> using
namespace std;

class student
{
    char name[100];
int age,roll;     float
percent;     public:
void getdata()
        {
              cout<<"Enter data"<<endl;
cout<<"Name:";            cin>>name;
cout<<"Age:";            cin>>age;
cout<<"Roll:";            cin>>roll;
cout<<"Percent:";
cin>>percent;         cout<<endl;
    }
    student & max(student &s1,student &s2)
    {
        if(percent>s1.percent && percent>s2.percent)
return *this;
        else if(s1.percent>percent && s1.percent>s2.percent)
return s1;
        else if(s2.percent>percent && s2.percent>s1.percent)
return s2;
    }
    void display()
    {
        cout<<"Name:"<<name<<endl;
cout<<"Age:"<<age<<endl;         cout<<"Roll:"<<roll<<endl;
cout<<"Percent:"<<percent;
    }
};

int main() {
  student s,s1,s2,s3;
s1.getdata();   s2.getdata();
s3.getdata();
s=s3.max(s1,s2);
  cout<<"Student with highest percentage"<<endl;
s.display();   getch();   return 0; }
```

Output

```
Enter data
Name:Paul
Age:24
Roll:11
Percent:79
```

18. **Program illustrate the use of pointers to objects whch are related by inheritance.** #include<iostream>

```
using namespace std;

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;

}
```

Output:

```
Bptr points to the base object
B = 100
Bptr now points to derived object
B =200
Dptr is  derived type pointer
B = 200
D = 300
Using((DC *)bptr)
B = 200 D
=400
```

19. **program illustrating the use of virtual functions in class.**

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

Output :

```
Bptr points to Base
Display Base
```

```
Show Base

Bptr pointd to derived

Display Base
Show derived
```

20. **program 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> using
namespace std;
#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;
}
```

Output:

```
Enter Book Details
Title : C++
Price: 220
Pages:500
Enter the Tape Details
Title: C++
Price : 220
Play Times(mins):45
Media Details
......................Book...........
Title: C++
Price: 220
Pages: 500
.....................Tape............
Title: C++
Play time: 45mins
Price: 220
```

21. **write a program show conversion from string to int and vice-versa.** #include<iostream>

```
#include <boost/lexical_cast.hpp>// for lexical_cast()
```

```
#include <string> // for string  using
namespace std;  int main()  {
  string str = "5";      string str1 =
"6.5";

  // Initializing f_value with casted float
  // f_value is 6.5
  float f_value = boost::lexical_cast<float>(str1);

  // Initializing i_value with casted int
  // i_value is 5
  int i_value = boost::lexical_cast<int>(str);

  //Displaying casted values
  cout << "The float value after casting is : ";       cout << f_value
<<endl;
  cout << "The int value after casting is : ";       cout << i_value
<<endl;

  return 0;  }
```

Output:

```
The float value after casting is : 6.5
The int value after casting is : 5
```

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

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

# include <stdlib.h>
 class date
{
    int mm,dd,yy;
public:     date()
{     }
    date(char dt[])
    {
        dd=getdt(dt,0);          mm=getdt(dt,3);
yy=getdt(dt,6);
    }      int getdt(char *,int);
void display();
};  main()
{ date
d1; char
dt[10];

cout<<"ENTER THE DATE IN DD/MM/YY
FORMAT:->"; cin>>dt; d1=date(dt); d1.display();
}
int date::getdt(char dt[],int i)
{
    char temp[2];
temp[0]=dt[i];
temp[1]=dt[i+1];
temp[2]='\0';     return atoi
(temp);
}
void date::display()
{
    cout<<"DAY= "<<dd<<endl;
cout<<"\nMONTH= "<<mm<<endl;
cout<<"\nYEAR= "<<yy<<endl;
} Output:
ENTER THE DATE IN DD/MM/YY FORMAT:-> 12/12/1998
DAY = 12
```

```
MONTH = 12
YEAR = 1998
```

24. **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> using
namespace std;
#include<fstream>
#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;  }
```

Output

```
Enter a String Abhishek
```

25. **copy the contents of one file to another.**

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

int main( )  {
```

```
char source[ 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 );
}

}
```

```
Output :
Enter the file name
Enter the target file name
```

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

```
#include<iostream>
using namespace std;
#include<fstream>  int
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;
}
}
```

Output

```
Enter a record 25
24
16
14
Add another Y/N
N
25 24 16 14
```

27.

27.files. maintain a elementary database of employees using

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


#include<conio.h>
#include<fstream>
#include<stdlib.h>
#include<iomanip>
#include<string.h> #include<stdio.h>
const char* fileName="Employee.txt";

class Employee
{ private:
int EmpID;
char
EmpName[
50],Post[50
],Departme
nt[10];
float
Salary;
public:
void
ReadData()
; int
GetID();
void
DisplayRec
ord(); char*
GetDepart
ment();
```

```
};
void Employee::ReadData()
{
cout<<endl<<"Employee ID:";
cin>>EmpID;
cout<<"Employee Name:";
cin>>EmpName;
cout<<"Employee's Post:";
cin>>Post;
cout<<"Employee's Department:";
cin>>Department; cout<<"Salary:";
cin>>Salary;
}
void Employee::DisplayRecord()
{
cout<<endl<<"_______________________________";
cout<<endl<<setw(5)<<EmpID<<setw(15)<<EmpName<<setw(15)<<
Pos t<<setw(15)<<Department<<setw(8)<<Salary;
}
int Employee::GetID()
{
return EmpID;
}
char* Employee::GetDepartment()
{
return Department;
}
  int main()
{
Employee emp,e; char
option,ch,Dept[50]; int
ID,isFound; clrscr();
fstream file;
```

```
file.open(fileName,ios::ate|ios::in|ios::out|ios::binary); do {
cout<<"*******Menu********"; cout<<endl<<"Enter your option";
cout<<endl<<"1 => Add a new record"; cout<<endl<<"2 => Search
record from employee id"; cout<<endl<<"3 => List Employee of
particular department"; cout<<endl<<"4 => Display all
employee"; cout<<endl<<"5 => Update record of an employee";
cout<<endl<<"6 => Delete record of particular employee";
cout<<endl<<"7 => Exit from the program"<<endl;
cout<<"********************"<<endl; cin>>option; switch(option)
{ case '1':
emp.ReadData(); file.seekg(0,ios::beg);
isFound=0;
file.read((char*)&e,sizeof(e)); while(!file.eof())
{
if(e.GetID()==emp.GetID())
{
cout<<"This ID already exist...Try for another ID";
isFound=1; break; }
file.read((char*)&e,sizeof(e));
}
if(isFound==1) break; file.clear();
file.seekp(0,ios::end);
file.write((char*)&emp, sizeof(emp));
cout<<endl<<"New record has been added successfully...";
break;   case '2': isFound=0;
cout<<endl<<"Enter ID of an employee to be searched:";
cin>>ID;
file.seekg(0,ios::beg); file.read((char*)&e,sizeof(e));
while(!file.eof())
{
if(e.GetID()==ID)
{
cout<<endl<<"The record found...."<<endl;
```

```
cout<<endl<<setw(5)<<"ID"<<setw(15)<<"Name"<<setw(15)<<"Post
"<<setw(15)<<"Department"<<setw(8)<<"Salary";
e.DisplayRecord();
isFound=1; break; }
file.read((char*)&e,sizeof(e));
} file.clear();
if(isFound==0)
cout<<endl<<"Data not found for employee ID#"<<ID;
break; case '3': isFound=0;
cout<<"Enter department name to list employee within it:";
cin>>Dept; file.seekg(0,ios::beg); file.read((char*)&e,sizeof(e));
while(!file.eof())
{
if(strcmp(e.GetDepartment(),Dept)==0)
{
cout<<endl<<"The record found for this department"<<endl;

cout<<endl<<setw(5)<<"ID"<<setw(15)<<"Name"<<setw(15)<<"Post
"<<setw(15)<<"Department"<<setw(8)<<"Salary";
e.DisplayRecord();
isFound=1; break; }
file.read((char*)&e,sizeof(e));
} file.clear();
if(isFound==0)
cout<<endl<<"Data not found for department"<<Dept;
break;   case '4': cout<<endl<<"Record for employee";
file.clear(); file.seekg(0,ios::beg); int counter=0;
file.read((char*)&e,sizeof(e)); while(!file.eof())
{ counter++;
if(counter==1)
{
cout<<endl<<setw(5)<<"ID"<<setw(15)<<"Name"<<setw(15)<<"Post
"<<setw(15)<<"Department"<<setw(8)<<"Salary";
```

```
}
e.DisplayRecord();
file.read((char*)&e,sizeof(e));
}
cout<<endl<<counter<<"records found......";
file.clear(); break;   case '5':
int recordNo=0;
cout<<endl<<"File is being modified...."; cout<<endl<<"Enter
employee ID to be updated:";
cin>>ID; isFound=0;
file.seekg(0,ios::beg);
file.read((char*)&e,sizeof(e));
while(!file.eof())
{ recordNo++;
if(e.GetID()==ID)
{
cout<<"The old record of employee having ID"<<ID<< "is:";
e.DisplayRecord();
isFound=1; break; }
file.read((char*)&e,sizeof(e));
}
if(isFound==0)
{
cout<<endl<<"Data not found for employee ID#"<<ID;
break; } file.clear();
int location=(recordNo-1)*sizeof(e); file.seekp(location,ios::beg);
cout<<endl<<"Enter new record for employee having ID"<<ID;
e.ReadData();
file.write((char*)&e, sizeof(e)); break;
case '6': recordNo=0;
cout<<endl<<"Enter employment ID to be deleted:";
cin>>ID; isFound=0; file.seekg(0,ios::beg);
file.read((char*)&e,sizeof(e)); while(!file.eof())
```

```
{ recordNo++;
if(e.GetID()==ID)
{ cout<<" The old record of employee having ID "<<ID<< " is:
";
e.DisplayRecord();
isFound=1; break; }
file.read((char*)&e,sizeof(e));
}
char tempFile[]="temp.txt";
fstream temp(tempFile,ios::out|ios::binary); if(isFound==0)
{
cout<<endl<<"Data not found for employee ID#"<<ID;
break; } else { file.clear(); file.seekg(0,ios::beg);
file.read((char*)&e,sizeof(e)); while(!file.eof())
{
if(e.GetID()!=ID)
temp.write((char*)&e,sizeof(e)); file.read((char*)&e,sizeof(e));
} file.close();
temp.close();
temp.open(tempFile,ios::in|ios::binary);
file.open(fileName,ios::out|ios::binary);
temp.read((char*)&e,sizeof(e)); while(!temp.eof())
{
file.write((char*)&e,sizeof(e)); temp.read((char*)&e,sizeof(e));
} } temp.close();
file.close();
remove(tempFile);
file.open(fileName,ios::ate|ios::in|ios::out|ios::binary); break;
case '7': exit(0); break;   default: cout<<"Invalid Options";
}
cout<<"\nDo you want to continue.....?y/n";
cin>>ch; }while(ch!='n');
}
```

```
Output
**********Menu*****
Enter Your Option
1 => Add a new Record
2=> Search record from employ id
3 => List Employee of particular department 4 =>
Display all employee
5  => Update record of an employee
6  => Delete record of particular employee
7  => Exit from employee
*******************
```

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

```
#include<iostream> using
namespace std;
#include<fstream>
#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(int
i=1;
i<argc;
i++)
{
fin.open(argv[i]);
cout<<"Contents of "<<argv[i]<<"\n";
do { fin.get(ch); cout<<ch; }while(fin); cout<<"\n\n";  fin.close();
}    return
0;  }
```

Output

```
Argc = 1
Error in arguments
```

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

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

Output

```
Stack is full
Item popped = 20
Item popped = 19
Item popped = 18
Item popped = 17
```

30. **program to implement a queue class with requried operations/ functions.**

```
// CPP program to implement Queue using
// two stacks with costly deQueue()
#include    <bits/stdc++.h>       using
namespace   std;     struct   Queue   {
stack<int> s1, s2;

   // Enqueue an item to the queue      void
enQueue(int x)      {
       // Push item into the first stack
s1.push(x);     }

   // Dequeue an item from the queue      int
deQueue()      {
       // if both stacks are empty          if
(s1.empty() && s2.empty()) {             cout <<
"Q is empty";              exit(0);         }
```

```
        // if s2 is empty, move          //
elements from s1          if (s2.empty()) {
while (!s1.empty()) {
s2.push(s1.top());                  s1.pop();
            }
        }

        // return the top item from s2          int x
= s2.top();          s2.pop();          return x;     }
};

//  Driver  code     int
main()
{
    Queue q;
    q.enQueue(1);
    q.enQueue(2);
    q.enQueue(3);

    cout << q.deQueue() << '\n';      cout
<< q.deQueue() << '\n';      cout <<
q.deQueue() << '\n';

    return 0;  }
```

Output

```
1 2 3
```

31. **a program to implement circular queue class with required operations/ functions.**

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

}
```

Output

```
Queue is full
Item deleted = 11
Item deleted = 12
Item deleted = 13
```

33. **implementing stack & its operations using dynamic memory allocation.**

```
 #include<iostream>
using namespace std;  #
include <process.h> #
include <conio.h>
# include <malloc.h>

struct link_list
{  int no;
 struct link_list *next;
}; class
stack {
 link_list *list,*head;
 public:     stack()
{      head=NULL;
free(list);     }
void push();
void peep();
void pop();
};
void stack :: push()
{   link_list *newnode;
newnode=new link_list;
cout<<"Enter Number to push :";
cin>>newnode->no;   list=head;
if(head==NULL)   {     head=newnode;
newnode->next=NULL;
  }   else   {     newnode-
>next=list;
list=newnode;
head=newnode;
  }
}
void stack :: peep()
```

```
{  cout<<endl;
if(head==NULL)  {
cout<<"Empty Stack !!!";
return;  }
 list=head;




   while(list!=NULL)  {
cout<<list->no<<"\t";
list=list->next;
 }
}
void stack :: pop()
{  cout<<endl;
if(head==NULL)  {
cout<<"Empty Stack !!!";
return;  }  else  {
  cout<<"Top : "<<head->no<<endl;   head=head->next;
 }
}  int display_menu() {  int
ch;  clrscr();  cout<<endl;
cout<<"[ 1 ] Push"<<endl;
cout<<"[ 2 ] Pop"<<endl;
cout<<"[ 3 ] Peep"<<endl;
cout<<"[ 4 ] Exit"<<endl;
cout<<"Enter your choice :";
cin>>ch;  return ch; } void
main() {  stack s1;  while(1)  {
  switch(display_menu())
  {    case 1:s1.push();
break;    case
2:s1.pop();       getch();
break;    case
3:s1.peep();
getch();       break;
case 4: exit(1);
  }
 }
} Output
:
[1] Push [2]
Pop
[3]  Peep
[4]  Exit
Enter your choice :1 Enter
number to push : 60
```

36. **multiple *catch* statements.**

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

Output:

```
Testing multiple catches
X==1
Caught an Integer
End of try-catch system
X==0
Caught a character
End of try-catch system
X==2
End of try-black
End of try-catch system
```

37. **rethrowing in exception.**

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

Output

```
Inside Main
Inside Function
Division  = 5.25
End of function
Inside Function
Caught double inside function
Caught double inside main End
of Main
```

38. **the functionality of testing the *throw* restrictions.**

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

Output:

```
Testining throw Exception
X== 0
Terminating called after throwing an instance of ‘char’
```

40. **write a program implementing stack and it’s operations using template class.**

```
#include<iostream> using
namespace std;
#include<conio.h>
#include<stdlib.h>
```

```
template<class Type>
class Stack
{ Type s[10];
int top,n;
public:
Stack() {
top=-1;
cout<<"\n\tEnter the Stack Size : ";
cin>>n; }
void push(Type elt)
{ if(top<n-1)
s[++top]=elt;
else
cout<<"\n\tstack is full.Can't insert "<<elt<<endl; }
void pop() {
if(top<0)
cout<<"\n\tstack is empty.\n"; else
cout<<"\n\tPoped elt : "<<s[top--];
}
void stk_operation();
};
template<class Type>
void Stack<Type> :: stk_operation()
{
int choice=1,i; Type
elt;
while(choice>0 && choice<3)
{
cout<<"\n\n\t1.PUSH\t2.POP\tAny Key To Exit\n\tChoice : ";
cin>>choice; switch(choice)
{ case 1 :
//push
cout<<"\n\tEnter the Elt to push : ";
cin>>elt; push(elt);
cout<<"\n\t\tstack content :\n\n\t";
for(i=0;i<=top;i++) cout<<s[i]<<"\t";
break; case 2 : //pop pop();
cout<<"\n\t\tstack content :\n\n\t";
for(i=0;i<=top;i++) cout<<s[i]<<"\t";
break; }
}}  int
main()
{
 cout<<"\n\t\tSTACK OPERATION USING
TEMPLATE\n\n";
cout<<"\n\t INT\n";
Stack<int> stk1;
cout<<"\n\t FLOAT\n";
Stack<float> stk2; int ch;
while(1) {
cout<<"\n\t\t\tSTACK OPERATION \n\n";
cout<<"\t1.INT STACK\t2.FLOAT STK\tAny Key To Exit\n\tChoice
: "; cin>>ch;
switch(ch) {
case 1 : //perform stk operation on int stk
stk1.stk_operation(); break; case 2 : //float
stk2.stk_operation(); break;
default : exit(0);
}
}
}
```

Output

```
  STACK OPERATION USING TEMPLATE
  INT
  Enter the size of Stack
```

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

```
#include<conio.h>
template<class T>
class node
{      public:
T data;
node<T> *link; };
template<class T>
class list {
private:
              node<T>
*first;       public:
list();       ~list();
void createlist();
void deletion();
void insertion();
      void display()
      };
template<class T>
list<T>::list()
{
               first=NULL;
}
template<class T>
list<T>::~list() {
node<T> *next;
                while(first)
{
next=first->link;
delete first;
first=next;
                }
```

```
}
template<class T>
void list<T>::createlist()
{
                    T a;
                    node<T> *cur,*ptr;
   cout<<"\nEnter data Of New Node (Enter 0 To Have other
options):";                     cin>>a;                     while(a)
                    {
                            cur=new
node<T>;                                cur-
>data=a;                                cur-
>link=NULL;
if(first==NULL)
first=cur;                              else
ptr->link=cur;
ptr=cur;
  cout<<"\nEnter data Of New Node (Enter 0 To Have other
options):";                             cin>>a;
                    }
}
template<class T>
void list<T>::insertion()
{
               node<T> *cur,*ptr;
                T ele;
char ch;
                ptr=first;
cur=new node<T>;
                cout<<"\nEnter data Of New
Node:";                 cin>>cur->data;
cur->link=NULL;
```

```
                cout<<"\n Do u wish to insert at the start [y/n]:";
cin>>ch;
                if(ch=='Y'||ch=='y')
                {
                                cur->link=first;
first=cur;
                }
else
                {
              cout<<"\n Specify after which element do u want to
insert :";                     cin>>ele;                    while(ptr!=NULL)
                    {
                                if(ptr->data==ele)
                                {
                                                cur-
>link=ptr->link;
ptr->link=cur;
break;
                                }
else
{
                                    ptr=ptr->link;
                                }
                    }
                }
}
template<class T>
void list<T>::deletion()
{
                T ele;
                char ch;
                node<T> *ptr,*ptr1;
                if(first==NULL)
                {
                        cout<<"\nSorry list is empty.";
                }
else
{
                  ptr=first;
                  cout<<"\nDo u want to delete first element? [y/n]:";
cin>>ch;
                  if(ch=='y'||ch=='Y')
                  {
                            first=first->link;
delete ptr;
                  }
else
                  {
                    cout<<"\nSpecify which element do u want to
delete :";                  cin>>ele;
                    while(ptr!=NULL)
                    {
                            if(ptr->link->data==ele)
                            {
                                        ptr1=ptr-
>link;                                       ptr-
>link=ptr1->link;
delete ptr1;
return;

}
else
{
                              ptr=ptr->link;
                            }
                    }
                    }
                }
}
template<class T>
void list<T>::display()
{
     node<T> *ptr;
     if(first==NULL)
     {
                    cout<<"\n Sorry list is empty..";
     }      else      {
ptr=first;
while(ptr!=NULL)
         {
             cout<<ptr->data<<"   ";
ptr=ptr->link;
         }
     }
}
int main()
{     int n;
list <int> l;
    l.createlist();
do
    {
                   cout<<"\n 1.Insertion \n2.Deletion \n3.Print List
\n4.Exit \n";                    cout<<"\n Enter your option : ";
cin>>n;
                   switch(n)

                   {
                            case 1: l.insertion();
break;
```

```
                        case 2: l.deletion();
break;
                        case 3: l.display();
                        break;
case 4:
exit(0);
break;
                }

}while(n<=4);
_getch();
return 0; }
```

Output:

```
Enter data of new node <Enter 0 To Have other options> :5
Enter data of new node <Enter 0 To Have other options> :3
Enter data of new node <Enter 0 To Have other options> :6
Enter data of new node <Enter 0 To Have other options> :7
Enter data of new node <Enter 0 To Have other options> :10
Enter data of new node <Enter 0 To Have other options> :0
```

- Insertion
- Deletion
- Print List
- Exit

Enter your option…