

**BANGALORE INSTITUTE OF TECHNOLOGY  
DEPARTMENT OF  
INFORMATION SCIENCE & ENGINEERING  
(Visveswaraya Technological University, Belgaum)**



**OOPS LAB MANUAL**

For  
VI Semester as per new syllabus 2006-07

**Prepared By:**  
**Shivakumar B.R., M.Tech**  
Lecturer, ISE Dept, BIT

**Vani.V, M.Tech**  
Lecturer, ISE Dept, BIT

**OBJECT ORIENTED PROGRAMMING LABORATORY**  
(Common to CSE & ISE)

Sub Code : 06CSL47  
Hrs / Week : 03  
Total Hrs : 42

IA Marks : 25  
Exam Hours : 03  
Exam Marks : 50

1. Given that an **EMPLOYEE** class contains following members: Data members : Employee\_Number, Employee\_Name, Basic, DA, IT, Net\_Salary Member functions: to read the data, to calculate Net\_Salary and to print data members. Write a C++ program to read the data of N employees and compute Net\_Salary of each employee. (Dearness Allowance (DA) = 52% of Basic and Income Tax (IT) = 30% of the gross salary.  $\text{Net\_Salary} = \text{Basic} + \text{DA} - \text{IT}$ )

Page No. 1-3

2. Define a **STUDENT** class with USN, Name, and Marks in 3 tests of a subject. Declare an array of 10 **STUDENT** objects. Using appropriate functions, find the average of two better marks for each student. Print the USN, Name, and the average marks of all the students.

Page No. 4-8

3. Write a C++ program to create a class called **COMPLEX** and implement the following overloading functions **ADD** that return a **COMPLEX** number.

i. **ADD** (a, s2) – where a is an integer (real part) and s2 is a complex number.

ii. **ADD** (s1, s2) – where s1 and s2 are complex numbers.

Page No. 9-11

4. Write a C++ program to create a class called **LIST** (linked list) with member functions to insert an element at the front of the list as well as to delete an element from the front of the list. Demonstrate all the functions after creating a list object.

Page No. 12-15

5. Write a C++ program to create a template function for Quick sort and demonstrate sorting of integers and doubles.

Page No. 16-19

6. Write a C++ program to create a class called **STACK** using an array of integers. Implement the following operations by overloading the operators + and -.

i.  $s1 = s1 + \text{element}$ ; where s1 is an object of the class **STACK** and element is an integer to be pushed on to top of the stack.

ii.  $s1 = s1 -$ ; where s1 is an object of the class **STACK** and - operator pops the element. Handle the **STACK Empty** and **STACK Full** conditions.

Also display the contents of the stack after each operation, by overloading the operator <<.

Page No. 20-23

7. Write a C++ program to create a class called DATE. Accept two valid dates in the form dd/mm/yy. Implement the following operations by overloading the operators + and -. After every operation display the results by overloading the operator <<.

i.  $\text{no\_of\_days} = d1 - d2$ ; where  $d1$  and  $d2$  are DATE objects,  $d1 \geq d2$  and  $\text{no\_of\_days}$  is an integer.

ii.  $d2 = d1 + \text{no\_of\_days}$ ; where  $d1$  is a DATE object and  $\text{no\_of\_days}$  is an integer.

Page No. 24-29

8. Write a C++ program to create a class called MATRIX using a two-dimensional array of integers. Implement the following operations by overloading the operator = which checks the compatibility of two matrices  $m1$  and  $m2$  to be added and subtracted. Perform the addition and subtraction by overloading the operators + and - respectively. Display the results (sum matrix  $m3$  and difference matrix  $m4$ ) by overloading the operator <<.

```
if(m1 == m2)
```

```
{
```

```
    m3 = m1 + m2;
```

```
    m4 = m1 - m2;
```

```
}
```

```
else
```

```
display error
```

Page No. 30-33

9. Write a C++ program to create a class called OCTAL, which has the characteristics of an octal number. Implement the following operations by writing an appropriate constructor and an overloaded operator +.

i.  $\text{OCTAL } h = x$ ; where  $x$  is an integer

ii.  $\text{int } y = h + k$ ; where  $h$  is an OCTAL object and  $k$  is an integer.

Display the OCTAL result by overloading the operator << Also display the values of  $h$  and  $y$ .

Page No. 34-35

10. Write a C++ program to create a class called QUEUE with member functions to add an element and to delete an element from the queue. Using these member functions, implement a queue of integers and doubles. Demonstrate the operations by displaying the

contents of the queue after every operation.

Page No. 36-40

11. Write a C++ program to create a class called DLIST (Doubly Linked List) with member functions to insert a node at a specified position and delete a node from a specified position of the list. Demonstrate the operation by displaying the contents of the list after every operation.

Page No. 41-46

**12. Write a C++ program to create a class called STUDENT with data members USN, Name and Age. Using inheritance, create the classes UGSTUDENT and PGSTUDENT having fields as Semester, Fees and Stipend. Enter the data for at least 5 students. Find the semester wise average age for all UG and PG students separately.**

Page No. 47-51

**13. Write a C++ program to create a class called STRING and implement the following operations. Display the results after every operation by overloading the operator <<.**

**i. STRING s1 = "VTU"**

**ii. STRING s2 = "BELGAUM"**

**iii. STIRNG s3 = s1 + s2 ; (Use copy constructor).**

Page No. 52-53

**14. Write a C++ program to create a class called BIN\_TREE ( Binary tree) with member functions to perform inorder, preorder and postorder traversals. Create a BIN\_TREE object and demonstrate the traversals.**

Page No. 54-57

**15. Write a C++ program to create a class called EXPRESSION. Using appropriate member functions convert a given valid Infix expression into Postfix form. Display the Infix and Postfix expressions.**

Page No. 58-61

**Note: In the examination *each* student picks one question from a lot of *all* 15 questions.**

**PROGRAM 1**

**Given that an employee class contains the following members: employee\_number, employee\_name, basic DA, IT, Net\_sal. member functions: to read the data, to calculate Net\_sal and to print data members. Write a C++ program to read the data of N employees and compute Net\_sal of each employee. DA=52% of basic IT=30% of gross salary.**

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

/* declaring a class */
class employee
{
    private:
        int num;
        char name[10];
        float basic, DA, gross, IT, netsal;
    public:
        void read_data();
        void calculate_n();
        void print_data();
};

/* function to read employee data */
void employee::read_data()
{
    cout << "enter the employee's number: ";
    cin >> num;
    cout << "enter the employee's name: ";
    cin >> name;
    cout << "enter the employee's basic salary: ";
    cin >> basic;
}

/* function to calculate the DA and IT */
void employee::calculate_n()
{
    DA = (float)(basic * 52 / 100);
    gross = DA + basic;
    IT = (float)(gross * 30 / 100);
    netsal = gross - IT;
}

/* function to print employee data */
```

```
void employee::print_data()
{
    cout<<num<<"\t"
        <<name<<"\t"
        <<basic<<"\t"
        <<DA<<"\t"
        <<IT<<"\t"
        <<netsal<<"\n";
}

/* main function */
int main()
{
    employee e1[10];
    int n,i;
    clrscr();
    cout<<"enter the number of employee's"<<endl;
    cin>>n;
    for (i=0;i<n;i++)
    {
        cout<<"enter"<<i+1<<" employee's data"<<endl;
        e1[i].read_data();
        e1[i].calculate_n();
    }
    cout<<"-----\n"
        <<"number   name   bas_salDA   IT   NET\n"
        <<"-----\n";
    for(i=0;i<n;i++)
    {
        e1[i].print_data();
    }
    cout<<"-----\n";

    return 0;
}
```

**OUTPUT:**

```
enter the number of employee's
3
enter1 employee's data
enter the employee's number:1
enter the employee's name:ram
enter the employee's basic salary:1000
enter2 employee's data
enter the employee's number:2
enter the employee's name: sita
enter the employee's basic salary:1500
enter3 employee's data
enter the employee's number:3
enter the employee's name: rita
enter the employee's basic salary:2000
-----
number name  bas_sal DA   IT   NET
-----
1   ram   1000  520  456  1064
2   sita  1500  780  684  1596
3   rita  2000  1040 912  2128
-----
Press any key to continue
```

.....

**PROGRAM 2**

**Define a STUDENT class with USN, Name and Marks in 3 tests of a subject. Declare an array of 10 STUDENT objects. Using appropriate functions, find the average of two better marks for each student. Print the USN, Name and the average marks of all the students.**

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

/* declaring a class */
class student
{
    private:
        int usn;
        char name[20];
        float m1,m2,m3;

    public:
        /* function to accept the data */
        void read()
        {
            cout<<"enter the student id   :";
            cin>>usn;
            cout<<"\nenter the student name   :";
            cin>>name;
            cout<<"\nenter the marks in 3 tests of a subject\n";
            cin>>m1>>m2>>m3;
        }

        /* function to find the average of two better marks */
        float average()
        {
            float lar,avg,slar;
            if(m1>=m2 && m1>=m3)
            {
                lar=m1;
                if(m2>=m3)
                    slar=m2;
                else
                    slar=m3;
            }
            if(m2>=m1 && m2>=m3)
            {
```

```

        lar=m2;
        if(m1>=m3)
            slar=m1;
        else
            slar=m2;
    }
    if(m3>=m1 && m3>=m2)
    {
        lar=m3;
        if(m2>=m1)
            slar=m2;
        else
            slar=m3;
    }
    avg=(float)((lar+slar)/2.0);
    return avg;
}

/* function display */
void display()
{
    cout<<usn<<setw(14)<<name<<setw(12)
        <<m1<<setw(14)<<m2<<setw(16)<<m3<<setw(20)
        <<average()<<endl;
}
}; /* end of class student */

/* main function */
void main()
{
    student s[20];
    int n;
    clrscr();
    cout<<"\nenter the number of students:";
    cin>>n;
    for(int i=0;i<n;i++)
    {
        cout<<"\nenter the details of the student "<<i+1<<"\n";
        s[i].read();
    }
    cout<<"\ndetails of the students are...\n";
    cout<<"-----\n";
    cout<<"number\t"<<setw(9)<<"name"<<setw(12)
        <<"marks1"<<setw(14)<<"marks2"<<setw(16)
        <<"marks3"<<setw(20)<<"average"<<endl;
}

```

```
        cout<<"-----\n";
        for(i=0;i<n;i++)
            s[i].display();
        cout<<"-----\n";
    }
```

.....

## OUTPUT

enter the number of students:10

enter the details of the student 1

enter the student id :1

enter the student name :raman

enter the marks in 3 tests of a subject

25

24

23

enter the details of the student 2

enter the student id :2

enter the student name :mohan

enter the marks in 3 tests of a subject

23

21

22

enter the details of the student 3

enter the student id :3

enter the student name :yuktha

enter the marks in 3 tests of a subject

25

20

21

enter the details of the student 4

enter the student id :4

enter the student name :roshan

enter the marks in 3 tests of a subject

22

24

19

enter the details of the student 5

enter the student id :5

enter the student name :bhavya

enter the marks in 3 tests of a subject

25

21

25

enter the details of the student 6

enter the student id :6

enter the student name :teena

enter the marks in 3 tests of a subject

21

23

24

enter the details of the student 7

enter the student id :7

enter the student name :suhas

enter the marks in 3 tests of a subject

22

20

25

enter the details of the student 8

enter the student id :8

enter the student name :priyanka

enter the marks in 3 tests of a subject

21

21

23

enter the details of the student 9

enter the student id :9

enter the student name :darshan

enter the marks in 3 tests of a subject

20

24

24

enter the details of the student 10

enter the student id :10

enter the student name :sumith

enter the marks in 3 tests of a subject

25

24

21

details of the students are...

```
-----  
number  name    marks1  marks2  marks3  average  
-----  
1      raman     25      24      23      24.5  
2      mohan     23      21      22      22.5  
3      yuktha    25      20      21      23  
4      roshan    22      24      19      23  
5      bhavya    25      21      25      25  
6      teena     21      23      24      23.5  
7      suhas     22      20      25      25  
8      priyanka  21      21      23      22  
9      darshan   20      24      24      24  
10     sumith    25      24      21      24.5  
-----
```

Press any key to continue

.....

**Program 3**

**Write a C++ program to create a class called COMPLEX and implement the following overloading functions ADD that return a COMPLEX number.**

- i) ADD(s1,s2)-where s1 is an integer(real part) and s2 is a complex number.**
- ii)ADD(s1,s2)-where s1 and s2 are complex numbers.**

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

/* Declaration of class */
class COMPLEX
{
    private:
        int realp,imagp;
    public:

        void readvalues()
        {
            cout<<"enter the real part and imaginary part\n";
            cin>>realp>>imagp;
        }

        friend COMPLEX add(COMPLEX,COMPLEX);
        friend COMPLEX add(int,COMPLEX);
        friend ostream & operator <<(ostream &,COMPLEX &);
};

/* function to add two complex numbers */
COMPLEX add(COMPLEX s1,COMPLEX s2)
{
    COMPLEX c;
    c.realp=s1.realp+s2.realp;
    c.imagp=s1.imagp+s2.imagp;
    return c;
}

/* function to add an integer and a complex number */
COMPLEX add(int ival,COMPLEX s2)
{
    COMPLEX c;
    c.realp=ival+s2.realp;
    c.imagp=s2.imagp;
    return c;
}
```

```
}

/* function to overload the << operator to display the complex numbers */
ostream & operator <<(ostream &out,COMPLEX &cmp)
{
    cout<<cmp.realp<<" +i"<<cmp.imagp<<endl;
    return out;
}

/* main program */
int main()
{
    int ival,choice;
    COMPLEX c1,c2,c3;
    clrscr();
    cout<<"1:to add an integer and a complex number\n";
    cout<<"2:to add two complex numbers\n";
    cout<<"enter your choice:";
    cin>>choice;

    switch(choice)
    {
        case 1:
            cout<<"enter the real part(integer):";
            cin>>ival;
            COMPLEX c4;
            c1.readvalues();
            cout<<"\nreal val="<<ival<<endl;
            cout<<" c1="<<c1;
            cout<<"-----\n";
            c4=add(ival,c1);
            cout<<" c4="<<c4;
            break;

            case 2:
                c1.readvalues();
                c2.readvalues();
                cout<<"\nc1="<<c1;
                cout<<"\nc2="<<c2;
                cout<<"-----\n";
                c3=add(c1,c2);
                cout<<"c3="<<c3;
                break;

            default:
                cout<<"invalid choice\n";
```

```
        break;

    }
    return 0;
    getch();
}
```

.....

## OUTPUT

RUN-1

```
1:to add an integer and a complex number
2:to add two complex numbers
enter your choice:1
enter the integer:4
enter the real part and imaginary part
3
8
```

Real val=4

c1=3+i8

-----

c4=7+i8

Press any key to continue

RUN-2

```
1:to add an integer and a complex number
2:to add two complex numbers
enter your choice:2
enter the real part and imaginary part
2
6
enter the real part and imaginary part
3
5
```

c1=2+i6

c2=3+i5

-----

c3=5+i11

.....

**PROGRAM 4**

**Write a C++ program to create a class called LIST(linked list) with member functions to insert an element at the front as well as to delete an element from the front of the list. Demonstrate all the functions after creating a list object.**

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

/* structure declaration*/
struct node
{
    int info;
    node *next;
};

/* class declaration*/
class list
{
    node *head;
public:
    /* constructor */
    list()
    {
        head=NULL;
    }

    /* Member function to insert an element to the linked list*/
    void insert()
    {
        int val;
        cout<<"enter the element:";
        cin>>val;
        node *newnode= new node;
        newnode->info=val;
        newnode->next=head;
        head=newnode;
    }

    /*Member function to delete an element from the linked list*/
```

```
void remove()
{
    node *temp=head;
    if(temp==NULL)
        cout<<"list is empty\n";
    else
    {
        cout<<"the item to be deleted is "<<temp->info<<endl;
        head=head->next;
        delete temp;
    }
}

/*Member function to display the contents of the linked list*/
void display()
{
    node *temp=head;
    while(temp!=NULL)
    {
        cout<<temp->info<<"-->";
        temp=temp->next;
    }
    cout<<"NULL\n";
}

};

/* main program */
void main()
{
    int choice;
    clrscr();
    list link;
    do
    {
        cout<<"\n\t MENU\n1:insert at front  2:delete at front"
            <<"\n3:display      4:exit\n";
        cout<<"enter your choice:";
        cin>>choice;

        switch(choice)
        {
```

```
        case 1:
            link.insert();
            break;

        case 2:
            link.remove();
            break;

        case 3:
            link.display();
            break;

        case 4:
            return;
    }
    //getch();
}while(choice<4);
}
```

---

## OUTPUT

```
MENU
1:insert at front  2:delete at front
3:display         4:exit
enter your choice:2
list is empty
```

```
MENU
1:insert at front  2:delete at front
3:display         4:exit
enter your choice:3
NULL
```

```
MENU
1:insert at front  2:delete at front
3:display         4:exit
enter your choice:1
enter the element:10
```

```
MENU
1:insert at front  2:delete at front
3:display         4:exit
enter your choice:1
enter the element:20
```

```
MENU
```

1:insert at front 2:delete at front  
3:display 4:exit  
enter your choice:1  
enter the element:30

MENU

1:insert at front 2:delete at front  
3:display 4:exit  
enter your choice:1  
enter the element:40

MENU

1:insert at front 2:delete at front  
3:display 4:exit  
enter your choice:1  
enter the element:50

MENU

1:insert at front 2:delete at front  
3:display 4:exit  
enter your choice:3  
50-->40-->30-->20-->10-->NULL

MENU

1:insert at front 2:delete at front  
3:display 4:exit  
enter your choice:2  
the item to be deleted is 50

MENU

1:insert at front 2:delete at front  
3:display 4:exit  
enter your choice:2  
the item to be deleted is 40

MENU

1:insert at front 2:delete at front  
3:display 4:exit  
enter your choice:3  
30-->20-->10-->NULL

.....

**PROGRAM-5**

**Write a C++ program to create a template function for QUICK SORT and demonstrate sorting of integers and doubles**

```
#include<iostream.h>
#include<conio.h>
#define SIZE 10

//Template declaration
template<class T>

//class declaration
class QSORT
{
    private:
        T array[SIZE];
        int len;

    public:
        QSORT (int);    /* constructor */
        void getarray();
        void quicksort(int,int);
        int partition(int,int);
        void printarray();
};

template<class T>

QSORT <T>::QSORT(int length)
{
    len=length;
    /* initialize all array elements to 0 */
    for(int k=0;k<len;k++)
        array[k]=0;
}

template<class T>
//Elements population to the array
void QSORT <T>::getarray()
{
    cout<<"enter the elements of the array\n";
    for(int i=0;i<len;i++)
        cin>>array[i];
}
template<class T>
```

**// Array sorting member functions**

```
void QSORT <T>:: quicksort(int low,int high)
```

```
{
    int pos;
    if(low<high)
    {
        pos=partition(low,high);
        quicksort(low,pos-1);
        quicksort(pos+1,high);
    }
}
```

```
template<class T>
```

```
int QSORT <T>:: partition(int low,int high)
```

```
{
    T key,temp;
    int left,right,tree=1;
    left=low;
    right=high;
    key=array[low]; /* assume that the first element is the pivot element */
    while(1)
    {
        while(left<high && key>=array[left])
            left++;
        while(key<array[right])
            right--;
        if(left<right)
        {
            temp=array[left];
            array[left]=array[right];
            array[right]=temp;
        }
        else
        {
            temp=array[low];
            array[low]=array[right];
            array[right]=temp;
            return(right);
        }
    }
}
```

```
template<class T>
```

```
// Member function for Displaying the printed array
```

```
void QSORT <T>:: printarray()
```

```
{
    cout<<"sorted array is...\n";
    for(int i=0;i<len;i++)
        cout<<array[i]<<endl;
}

//Main function
int main()
{
    int n,low,high;
    clrscr();
    cout<<"enter the size of the array:";
    cin>>n;

    low=0;
    high=n-1;

    QSORT <int> IA(n);
    QSORT <double> DA(n);

    cout<<"integer array processing...\n";
    IA.getarray();
    IA.quicksort(low,high);
    IA.printarray();

    cout<<"double array processing...\n";
    DA.getarray();
    DA.quicksort(low,high);
    DA.printarray();getch();
}
```

.....

**OUTPUT**

enter the size of the array:5

integer array processing...

enter the elements of the array

34

67

12

99

32

sorted array is...

12

32

34

67

99

double array processing...

enter the elements of the array

45.9

54.8

1.98

4.66

47.9

sorted array is...

1.98

4.66

45.9

47.9

54.8

Press any key to continue

.....

**PROGRAM -6**

**Write a C++ program to create a class called STACK using an array of integers. Implement the following operations by overloading the operators + and –**

**i) s1=s1+element; where s1 is an object of the class STACK and element is an integer to be pushed on the top of the stack.**

**ii) s1=s1-; where s1 is an object of the class STACK - operator pops the element. Handle the STACK empty and STACK full conditions. Also display the contents of the stack after each operation, by overloading the operator <<**

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

```
const int SIZE=5; //Stack size
```

```
//class declaration
```

```
class stack
```

```
{
```

```
    int items[SIZE];
```

```
    int top;
```

```
    int full();
```

```
    int empty();
```

```
    public:
```

```
        stack()
```

```
        {
```

```
            top=-1;
```

```
        }
```

```
        stack operator--(int);
```

```
        friend stack operator+(stack s1,int elem);
```

```
        friend ostream &operator<<(ostream &os,stack &s1);
```

```
};
```

```
// checking for Stack overflow
```

```
int stack::full()
```

```
{
```

```
    if(top==SIZE-1)
```

```
        return 1;
```

```
    else
```

```
        return 0;
```

```
}
```

**//Checking for stack under flow.**

```
int stack::empty()
{
    if(top== -1)
        return 1;
    else
        return 0;
}
```

**//function for element deletion from the stack**

```
stack stack::operator--(int )
{
    if(empty())
    {
        cout<<"Stack underflow\n";
    }
    else
    {
        cout<<"\nThe element deleted is :"  
        <<items[top];
        stack t;
        t.top=--top;
        for(int i=0;i<=top;i++)
            t.items[i]=items[i];
    }
    return *this;
}
```

```
ostream &operator<<(ostream &os,stack &s1)
{
    for(int i=s1.top;i>=0;i--)
        os<<s1.items[i]<<"\n";
    return os;
}
```

**//function for element insertion on to the stack**

```
stack operator+(stack s1,int elem)
{
    if(s1.full())
        cout<<"\nStack overflow\n";
    else
        s1.items[++(s1.top)]=elem;
    return s1;
}
```

```
/*Main function*/
```

```
void main()
```

```
{
    stack s1;
    int choice,elem;
    clrscr();
    for(;;)
    {
        cout<<"\n1:PUSH 2:POP 3:DISPLAY 4:EXIT\n"
            <<"enter your choice:";
        cin>>choice;
        switch(choice)
        {
            case 1:
                cout<<"Enter the element to be inserted:";
                cin>>elem;
                s1=s1+elem;
                break;
            case 2:
                s1=s1--;
                break;
            case 3:
                cout <<"The contents of the stack are :\n"
                    <<s1;
                break;

            case 4: exit(0);
            default:
                cout <<"Invalid choice\n";
                getch();
                exit(0);
        }
    }
}
```

---

## OUTPUT

```
1:PUSH 2:POP 3:DISPLAY 4:EXIT
enter your choice:2
Stack underflow
```

```
1:PUSH 2:POP 3:DISPLAY 4:EXIT
enter your choice:1
Enter the element to be inserted:20
```

```
1:PUSH 2:POP 3:DISPLAY 4:EXIT
```

enter your choice:1  
Enter the element to be inserted:45

1:PUSH 2:POP 3:DISPLAY 4:EXIT  
enter your choice:1  
Enter the element to be inserted:51

1:PUSH 2:POP 3:DISPLAY 4:EXIT  
enter your choice:1  
Enter the element to be inserted:62

1:PUSH 2:POP 3:DISPLAY 4:EXIT  
enter your choice:1  
Enter the element to be inserted:77

1:PUSH 2:POP 3:DISPLAY 4:EXIT  
enter your choice:3  
The contents of the stack are :  
77  
62  
51  
45  
20

1:PUSH 2:POP 3:DISPLAY 4:EXIT  
enter your choice:2  
The element deleted is :77

1:PUSH 2:POP 3:DISPLAY 4:EXIT  
enter your choice:2  
The element deleted is :62

1:PUSH 2:POP 3:DISPLAY 4:EXIT  
enter your choice:3  
The contents of the stack are :  
51  
45  
20

1:PUSH 2:POP 3:DISPLAY 4:EXIT  
enter your choice:4

**PROGRAM-7**

**Write a C++ program to create a class called DATE. Accept two valid dates in the form dd/mm/yy. Implement the following operations by overloading the + and - operators. After every operation display the results by overloading the operator <<.**

**i)no\_of\_days=d1-d2; where d1 and d2 are date objects, d1>=d2 and no\_of\_days is an integer.**

**ii)d2=d1+no\_of\_days; where d1 is a DATE object and no\_of\_days is an integer.**

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

//class declaration
class date
{
    private:
        int mm,dd,yy;

    public:
        void getdate();
        int operator -(date);
        date operator +(int);
};

// Member function to accept the date from the user
void date::getdate()
{
    cout<<"enter a valid date(dd mm yy)\n";

    START:
    cin>>dd>>mm>>yy;

    if((mm==2) && (dd>29))
    {
        cout<<"wrong input!!!\n";
        cout<<"\nenter the date again....\n";
        goto START;
    }

    if((mm>12)||(dd>31))
    {
        cout<<"wrong input!!!\n";
        cout<<"\nenter the date again....\n";
        goto START;
    }
}
```

```

    }

    if((mm==4||mm==6||mm==9||mm==11) && (dd>30))
    {
        cout<<"wrong input!!\n";
        cout<<"\nenter the date again...\n";
        goto START;
    }

    if((yy%4)!=0 && (mm==2) && (dd>28))
    {
        cout<<"wrong input!!\n";
        cout<<"\nenter the date again...\n";
        goto START;
    }
}

```

**// operator overloaded function to find the no. of days between two dates**

```

int date::operator -(date d2)
{
    int i,nod1,nod2,nody,lc,no_of_days;
    nod1=nod2=lc=0;

    for(i=1;i<mm;i++)
    {
        if(i==1||i==3||i==5||i==7||i==8||i==10||i==12)
            nod1+=31;
        else if(i==2)
            nod1+=28;
        else
            nod1+=30;
    }

    nod1+=dd;

    for(i=1;i<d2.mm;i++)
    {
        if(i==1||i==3||i==5||i==7||i==8||i==10||i==12)
            nod2+=31;
        else if(i==2)
            nod2+=28;
        else
            nod2+=30;
    }

    nod2+=d2.dd;
}

```

```

    nody=(yy-d2.yy)*365;

    for(i=d2.yy;i<yy;i++)
        if((i%4)==0)
            lc++;

    int y4=yy-d2.yy;
    while(y4>400)
    {
        lc++;
        y4-=400;
    }

    if((mm>2) && (yy%4)==0)
        lc++;
    if((d2.mm>2) && (d2.yy%4)==0)
        lc--;

    no_of_days=nody+nod1-nod2+lc;
    if(no_of_days>0)
    {
        cout<<"total number of days between these dates is=";
        return(no_of_days);
    }
    else
        return(no_of_days);
}

/* operator overloaded function to find the new date when no. of days are added to a particular date.*/

date date::operator +(int nd)
{
    date dd3;

    while(nd>365)
    {
        yy++;
        nd-=365;
    }

    while(nd>30)
    {
        if(mm==1||mm==3||mm==5||mm==7||mm==8||mm==10||mm==12)
        {
            nd-=31;

```

```
        mm++;
    }
    else if(mm==2)
    {
        nd-=28;
        mm++;
    }
    else
    {
        nd-=30;
        mm++;
    }

    if(mm>12)
    {
        yy++;
        mm=1;
    }
}

dd=dd+nd;
if(dd>30)
{
    if(mm==4||mm==6||mm==9||mm==11)
    {
        mm++;
        dd-=30;
    }
    else if(mm==2)
    {
        mm++;
        dd-=28;
    }
    else if(dd>31)
    {
        mm++;
        dd-=31;
    }
    if(mm>12)
    {
        yy++;
        mm=1;
    }
}

dd3.mm=mm;
```

```
        dd3.dd=dd;
        dd3.yy=yy;
        cout<<"new date is:";
        cout<<dd<<"- "<<mm<<"- "<<yy<<endl;
        return(dd3);
    }

//main function
void main()
{
    int res,num;
    date dd1,dd2;

    BEGIN:
        dd1.getdate();
        dd2.getdate();
        res=dd1-dd2;

    if(res<0)
        {
            cout<<"\nthe first date should be greater than the second date\n";
            cout<<"so enter the dates again\n";
            goto BEGIN;
        }

        cout<<res;
        cout<<"\nenter the no. of days to be added to the FIRST date:";
        cin>>num;

        dd2=dd1+num;
    }
```

.....

## OUTPUT

RUN-1

```
enter a valid date(dd mm yy)
30 2 2004
wrong input!!!
```

```
enter the date again....
```

```
29 2 2004
enter a valid date(dd mm yy)
12 6 2003
total number of days between these dates is=262
enter the no. of days to be added to the FIRST date:23
```

new date is:24-3-2004  
Press any key to continue

RUN-2  
enter a valid date(dd mm yy)  
31 12 2004  
enter a valid date(dd mm yy)  
31 12 2005

the first date should be greater than the second date  
so enter the dates again  
enter a valid date(dd mm yy)  
31 12 2004  
enter a valid date(dd mm yy)  
31 12 2003  
total number of days between these dates is=366  
enter the no. of days to be added to the FIRST date:13  
new date is:13-1-2005  
Press any key to continue

.....

**PROGRAM-8**

**Write a C++ program to create a class called MATRIX using a two dimensional array of integers. Implement the following operations by overloading the operator == which checks the compatibility of the two matrices to be added and subtracted. Perform the addition and subtraction by overloading the operators + and - respectively. Display the results by overloading operator <<.**

```

if(m1==m2)
    {
        m3=m1+m2;
        m4=m1-m2;
    }
else
    display error

```

```

class matrix
{
private:long m[5][5];
        int row;int col;
public:void getdata();
        int operator ==(matrix);
        matrix operator+(matrix);
        matrix operator-(matrix);
        friend ostream & operator << (ostream &,matrix &);
};

/* function to check whether the order of matrix are same or not */
int matrix::operator==(matrix cm)
{
    if(row==cm.row && col==cm.col)
    {
        return 1;
    }
    return 0;
}

/* function to read data for matrix*/
void matrix::getdata()
{
    cout<<"enter the number of rows\n";
    cin>>row;
    cout<<"enter the number of columns\n";
    cin>>col;
    cout<<"enter the elements of the matrix\n";

```

```
        for(int i=0;i<row;i++)
        {
            for(int j=0;j<col;j++)
            {
                cin>>m[i][j];
            }
        }
    }

/* function to add two matrix */
matrix matrix::operator+(matrix am)
{
    matrix temp;
    for(int i=0;i<row;i++)
    {
        for(int j=0;j<col;j++)
        {
            temp.m[i][j]=m[i][j]+am.m[i][j];
        }
        temp.row=row;
        temp.col=col;
    }
    return temp;
}

/* function to subtract two matrix */
matrix matrix::operator-(matrix sm)
{
    matrix temp;
    for(int i=0;i<row;i++)
    {
        for(int j=0;j<col;j++)
        {
            temp.m[i][j]=m[i][j]-sm.m[i][j];
        }
        temp.row=row;
        temp.col=col;
    }
    return temp;
}

/* function to display the contents of the matrix */
ostream & operator <<(ostream &fout,matrix &d)
{
    for(int i=0;i<d.col;i++)
```

```
        {
            for(int j=0;j<d.col;j++)
            {
                fout<<d.m[i][j];
                cout<<" ";
            }
            cout<<endl;
        }
        return fout;
    }

/* main function */
void main()
{
    matrix m1,m2,m3,m4;
    clrscr();
    m1.getdata();
    m2.getdata();
    if(m1==m2)
    {
        m3=m1+m2;
        m4=m1-m2;
        cout<<"Addition of matrices\n";
        cout<<"the result is\n";
        cout<<m3;
        cout<<"subtraction of matrices\n";
        cout<<"The result is \n";
        cout<<m4;
    }
    else
    {
        cout<<"order of the input matrices is not identical\n";
        exit(0);
    }
}
```

.....

**OUTPUT:**

enter the number of rows

2

enter the number of columns

2

enter the elements of the matrix

4

5

6

7

enter the number of rows

2

enter the number of columns

2

enter the elements of the matrix

9

8

7

6

Addition of matrices

the result is

13 13

13 13

subtraction of matrices

The result is

-5 -3

-1 1

Press any key to continue

.....

**PROGRAM-9**

**Write a c++ program to create a class called OCTAL which has characteristics of an octal number. Implement the following operations by writing an appropriate constructor & an overload operator**

**(1) octal n=x; x is an integer**

**(2) int y=n+k; n is an octal number &k is an integer**

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

/* declaring a class */
class octal
{
    int octnum;
    int conv_dec(int x);
public:
    octal(int x)
    {
        octnum=conv_dec(x);
    }
    int operator+(int k);
    friend ostream&operator<<(ostream&os,octal o);
};

/* function to convert a number in decimal to octal*/
int octal::conv_dec(int x)
{
    int i=0,sum=0;
    while(x!=0)
    {
        int rem=x%8;
        sum=sum+rem*pow(10,i);
        i++;
        x=x/8;
    }
    return sum;
}
```

```
/*function to perform y=n+k; n is an octal number &k is an integer*/
```

```
int octal::operator +(int x)
{
    x=conv_dec(x);
    return(octnum+x);
}

/*function to output octal value by overloading <<*/
ostream &operator <<(ostream &os,octal oct)
{
    os<<oct.octnum;
    return os;
}

/* main function */
void main()
{
    int x;
    clrscr();
    cout<<"enter the decimal number to be converted:"<<endl;
    cin>>x;
    octal n=x;
    cout<<"\noctal value is ="<<n;
    cout<<"\nenter value of k to be added:";
    cin>>x;
    int y=n+x;
    cout<<"\nvalue of y is ="<<y<<endl;
    getch();
}
```

.....

## OUTPUT

```
enter the decimal number to be converted:
10
```

```
octal value is =12
enter value of k to be added:12
```

```
value of y is =26
Press any key to continue */
```

**PROGRAM 10**

**Write a C++ program to create a class called QUEUE with member functions to add an element and to delete an element from the queue. Using these member functions, implement a queue of integer and double. Demonstrate the operations by displaying the content of the queue after every operation.**

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#define size 5

/* declaring a class */
template<class Q>class Queue
{
    Q que[size];
    int front,rear;
public:Queue() {front=0; rear=-1; }
    void Qinsert(Q ele);
    void Qdelete();
    void Qdisplay();
};

/* function to insert an element into the queue*/
template<class Q>void Queue<Q>::Qinsert(Q ele)
{
    if(rear==size-1)
        cout<<"\nQueue overflows";
    else
        que[++rear]=ele;
}

/* function to delete an element from the queue*/
template<class Q>void Queue<Q>::Qdelete()
{
    if(rear==-1)
        cout<<"\nQueue underflows";
    else
    {
        if(front==rear) /* Only one element is present*/
        {
            cout<<"\n"<<que[front]<<" deleted";
            front=0; rear=-1;
        }
    }
}
```

```

    }
    else
    {
        cout<<"\n"<<que[front]<<" deleted";
        front++;
    }
}
}

```

**/\* function to display the contents of the queue\*/**

```

template<class Q>void Queue<Q>::Qdisplay()
{
    if(rear== -1)
        cout<<"\nQueue is empty";
    else
    {
        cout<<"\nELements of Queue are:";
        for(int i=front;i<=rear;i++)
            cout<<que[i]<<" ";
    }
}

```

**/\* main function \*/**

```

void main(void)
{
    Queue<int>q1;
    Queue<double>q2;
    double x;
    int opt1=0,opt2=0,choice; clrscr();
    cout<<"\n1.Queue of integers\n2.Queue of doubles\n3.Exit\n";
    cout<<"\nEnter your choice:";
    cin>>choice;
    switch(choice)
    {
        case 1:while(opt1!=4)
            {
                cout<<"\n\t\tQueue of integers";
                cout<<"\n1.Insert\n2.Delete\n3.Display\n4.Exit\n";
                cout<<"\nEnter your option:";
                cin>>opt1;
                switch(opt1)
                {
                    case 1:cout<<"\nEnter the integer to be inserted:";
                        cin>>x;
                        q1.Qinsert(x);
                        getch();
                }
            }
    }
}

```

```
        break;
    case 2:q1.Qdelete();
        q1.Qdisplay();
        break;
    case 3:q1.Qdisplay();
        getch();
        break;
    }
}
getch();
break;
case 2:while(opt2!=4)
{
    cout<<"\n\tQueue of doubles";
    cout<<"\n1.Insert\n2.Delete\n3.Display\n4.Exit\n";
    cout<<"\nEnter your option:";
    cin>>opt2;
    switch(opt2)
    {
        case 1:cout<<"\nEnter the number to be inserted:";
            cin>>x;
            q2.Qinsert(x);
            break;
        case 2:q2.Qdelete();
            q2.Qdisplay();
            break;
        case 3:q2.Qdisplay();
            getch();
            break;
    }
}
getch();
break;
case 3:cout<<"\nProgram Terminated";
    getch();
    exit(0);
}
}
```

**OUTPUT**

1.Queue of integers  
2.Queue of doubles  
3.Exit  
Enter your choice:1

Queue of integers  
1.Insert  
2.Delete  
3.Display  
4.Exit  
Enter your option:1  
Enter the integer to be inserted:1

Queue of integers  
1.Insert  
2.Delete  
3.Display  
4.Exit  
Enter your option:1  
Enter the integer to be inserted:2

Queue of integers  
1.Insert  
2.Delete  
3.Display  
4.Exit  
Enter your option:1  
Enter the integer to be inserted:3

Queue of integers  
1.Insert  
2.Delete  
3.Display  
4.Exit  
Enter your option:3  
ELements of Queue are:1 2 3

Queue of integers  
1.Insert  
2.Delete  
3.Display  
4.Exit

Enter your option:2  
1 deleted  
ELements of Queue are:2 3

Queue of integers

1.Insert  
2.Delete  
3.Display  
4.Exit  
Enter your option:2  
2 deleted  
ELements of Queue are:3

Queue of integers

1.Insert  
2.Delete  
3.Display  
4.Exit  
Enter your option:2  
3 deleted  
Queue is empty

Queue of integers

1.Insert  
2.Delete  
3.Display  
4.Exit  
Enter your option:2  
Queue underflows  
Queue is empty

Queue of integers

1.Insert  
2.Delete  
3.Display  
4.Exit  
Enter your option:3  
Queue is empty

.....

**PROGRAM 11**

**Write a C++ program to create a class called DLIST (Doubly Linked List) with member functions to insert a node at a specified position and delete a node from a specified position of the list. Demonstrate the operations by displaying to content of the list after every operation.**

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

/* declaring a class */
class Dlist
{
    typedef struct nodetype
    {
        int info;
        struct nodetype *left,*right;
    }node;
    node *front;
public:
    Dlist() { front=NULL; }
    node* getnode()
    {
        node *ptr;
        ptr=(node*)malloc(sizeof(node));
        ptr->left=NULL;
        ptr->right==NULL;
        return ptr;
    }
    void insertfront(int x);
    void insert(int x,int ipos);
    void Delete(int dpos);
    void display();
};
/* function to insert an element in the front or beginning*/
void Dlist::insertfront(int x)
{
    node *p;
    p=getnode();
    p->info=x;
    p->right=front;
    front->left=p;
    front=p;
}
```

```
/* function to insert an element at a particular position*/
```

```
void Dlist::insert(int x,int ipos)
{
    node *p,*cur,*prev;
    int count=0;
    p=getnode();
    p->info=x;
    if(front==NULL)
        front=p;
    else if(ipos==1)
        insertfront(x);
    else
    {
        cur=front;
        prev=NULL;
        while(cur!=NULL && count!=ipos-1)
        {
            prev=cur;
            cur=cur->right;
            count++;
        }
        if(cur==NULL)
        {
            prev->right=p;
            p->left=prev;
        }
        else if(cur!=NULL)
        {
            prev->right=p;
            p->left=prev;
            cur->left=p;
            p->right=cur;
        }
        else if(prev==NULL)
            cout<<"\nInvalid position.";
    }
}
```

```
/* function to delete an element from a particular position*/
```

```
void Dlist::Delete(int dpos)
{
    node *cur;
    int count=0;
    if(front==NULL)
```

```

        cout<<"\nList underflows";
    else
    {
        if(dpos==1)
        {
            cur=front;
            cout<<"\n"<<front->info<<" deleted";
            front=front->right;
            free(cur);
        }
        else
        {
            cur=front;
            while(cur!=NULL && count!=dpos-1)
            {
                cur=cur->right;
                count++;
            }
            if(cur==NULL)
                cout<<"\nInvalid position";
            else
            {
                cout<<"\n"<<cur->info<<" deleted";
                cur->left->right=cur->right;
                cur->right->left=cur->left;
                free(cur);
            }
        }
    }
}

/* function to display the contents of the doubly linked list*/
void Dlist::display()
{
    node *temp;
    temp=front;
    if(temp==NULL)
    {
        cout<<"\nList is empty";
        getch();
    }
    while(temp!=NULL)
    {
        cout<<temp->info<<"->";
        temp=temp->right;
    }
}

```

```
    cout<<"NULL";
}

/* main function */
void main()
{
    Dlist d;
    int opt=0;
    while(opt!=4)
    {
        clrscr();
        cout<<"\n1.Insert\n2.Delete\n3.Display\n4.Exit\nEnter your option:";
        cin>>opt;
        switch(opt)
        {
            case 1:int x,ipos;
                cout<<"\nEnter the element to be inserted:";
                cin>>x;
                cout<<"\nEnter the position:";
                cin>>ipos;
                d.insert(x,ipos);
                getch();
                d.display();
                getch();
                break;
            case 2:int dpos;
                cout<<"\nEnter the position of deletion:";
                cin>>dpos;
                d.Delete(dpos); getch();
                d.display(); getch();
                break;
            case 3:cout<<"\nElements of the list are:";
                d.display(); getch();
                break;
            case 4:cout<<"\nProgram Terminated";
                getch();
                break;
        }
    }
}
```

.....

**OUTPUT**

```
1.Insert
2.Delete
3.Display
4.Exit
Enter your option:1
Enter the element to be inserted:6
Enter the position:1
6->NULL
```

```
1.Insert
2.Delete
3.Display
4.Exit
Enter your option:1
Enter the element to be inserted:10
Enter the position:2
6->10->NULL
```

```
1.Insert
2.Delete
3.Display
4.Exit
Enter your option:1
Enter the element to be inserted:4
Enter the position:2
6->4->10->NULL
```

```
1.Insert
2.Delete
3.Display
4.Exit
Enter your option:1
Enter the element to be inserted:3
Enter the position:3
6->4->3->10->NULL
```

```
1.Insert
2.Delete
3.Display
4.Exit
Enter your option:2
Enter the position of deletion:2
  4  deleted
```

6->3->10->NULL

- 1.Insert
- 2.Delete
- 3.Display
- 4.Exit

Enter your option:2

Enter the position of deletion:3

10 deleted

6->3->NULL

- 1.Insert
- 2.Delete
- 3.Display
- 4.Exit

Enter your option:2

Enter the position of deletion:1

4 deleted

3->NULL

.....

**PROGRAM 12**

**Write a C++ program to create a class called STUDENT with data members USN,Name and Age. Using inheritance, create the classes UGSTUDENT and PGSTUDENT having fields as Semester,Fees and Stipend. Enter the data for at least 5 students. Find the semester wise average age for all UG and PG students separately.**

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

/* declaring a class */
class student
{
    public:int reg,age;
        char name[20];
        void read_data();
};

class ugstudent:public student
{
    public:int stipend,sem;
        float fees;
        void read_data();
};

class pgstudent:public student
{
    public:int stipend,sem;
        float fees;
        void read_data();
};

/* function to read student details*/
void student::read_data()
{
    cout<<"\nEnter name:";
    cin>>name;
    cout<<"\nEnter Reg.no.";
    cin>>reg;
    cout<<"\nEnter age:";
    cin>>age;
```

```
}
/* function to read additional details for ugstudent*/
void ugstudent::read_data()
{
    student::read_data();
    cout<<"\nEnter the sem:";
    cin>>sem;
    cout<<"\nEnter the fees:";
    cin>>fees;
    cout<<"\nEnter the stipend:";
    cin>>stipend;
}

/* function to read additional details for pgstudents*/
void pgstudent::read_data()
{
    student::read_data();
    cout<<"\nEnter the sem:";
    cin>>sem;
    cout<<"\nEnter the fees:";
    cin>>fees;
    cout<<"\nEnter the stipend:";
    cin>>stipend;
}

/* main function */
void main()
{
    ugstudent ug[20];
    pgstudent pg[20];
    int i,n,m;
    float average; clrscr();
    cout<<"\nEnter the no. of entries in the ugstudent class:";
    cin>>n;
    for(i=1;i<=n;i++)
        ug[i].read_data();
    for(int sem=1;sem<=8;sem++)
    {
        float sum=0;
        int found=0,count=0;
        for(i=1;i<=n;i++)
        {
            if(ug[i].sem==sem)
            {
                sum=sum+ug[i].age;
                found=1;
            }
        }
    }
}
```

```
                count++;
            }
        }
        if(found==1)
        {
            average=sum/count;
            cout<<"\nAverage of age of sem "<<sem<<" is "<<average;

        }
    }
    cout<<"\nEnter the no. of entries of pgstudent class:";
    cin>>n;
    for(i=1;i<=n;i++)
        pg[i].read_data();
    for(sem=1;sem<=8;sem++)
    {
        float sum=0;
        int found=0,count=0;
        for(i=1;i<=n;i++)
        {
            if(pg[i].sem==sem)
            {
                sum=sum+pg[i].age;
                found=1;
                count++;
            }
        }
        if(found==1)
        {
            average=sum/count;
            cout<<"\nAverage of age of sem "<<sem<<" is "<<average;

        }
    }
    getch();
}
```

**OUTPUT**

Enter the no. of entries in the ugstudent class:3  
Enter name:Anagha  
Enter Reg.no.3  
Enter age:19  
Enter the sem:4  
Enter the fees:12000  
Enter the stipend:3489  
Enter name:Madhulika  
Enter Reg.no.17  
Enter age:20  
Enter the sem:4  
Enter the fees:15000  
Enter the stipend:3245  
Enter name:Ankitha  
Enter Reg.no.4  
Enter age:21  
Enter the sem:4  
Enter the fees:20000  
Enter the stipend:34278  
Average of age of sem 4 is 20  
Enter the no. of entries of pgstudent class:4  
Enter name:SnowWhite  
Enter Reg.no.6  
Enter age:19  
Enter the sem:4  
Enter the fees:78346  
Enter the stipend:478  
Enter name:Cindy  
Enter Reg.no.7  
Enter age:20  
Enter the sem:4  
Enter the fees:76479  
Enter the stipend:8734  
Enter name:Della  
Enter Reg.no.8  
Enter age:21  
Enter the sem:5  
Enter the fees:78578  
Enter the stipend:478  
Enter name:Stella  
Enter Reg.no.9  
Enter age:22  
Enter the sem:5  
Enter the fees:63278

Enter the stipend:748

Average of age of sem 4 is 19.5

Average of age of sem 5 is 21.5

.....

**PROGRAM 13**

**Write a C++ program to create a class called STRING and implement the following operations. Display the results after every operation by overloading the operator <<.**

- i)     STRING s1="VTU"**
- ii)    STRING s2="BELGAUM"**
- iii)   STRING s3=s1+s2; (Use copy constructor)**

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

/* declaring a class */
class string
{
    char str[20];
    public:string() { strcpy(str, " "); }
    string(char *str1) { strcpy(str,str1); }
    string operator+(string s2);
    friend ostream &operator<<(ostream &fout,string &s);
};

string string::operator+(string s2)
{
    strcat(str,s2.str);
    return str;
}

ostream &operator<<(ostream &fout,string &s)
{
    cout<<"\nThe string is:"<<s.str;
    return fout;
}

/* main function */
void main()
{
    char *str1,*str2; clrscr();
    cout<<"\nEnter string1:";
    gets(str1);
    cout<<"\nEnter string2:";
    gets(str2);
    string s1(str1);
```

```
    string s2(str2);
    string s3;
    cout<<"\nBefore concatenation:"<<endl;
    cout<<s1<<endl;
    cout<<s2<<endl;
    cout<<"\nAfter concatenation:"<<endl;
    s3=s1+s2;
    cout<<s3;
    getch();
    cout<<"\nProgram Terminated.";
    getch();
}
```

.....

### OUTPUT

```
Enter string1:Bangalore
Enter string2:Institute
Before concatenation:
The string is:Bangalore
The string is:Institute
After concatenation:
The string is:BangaloreInstitute
Program Terminated
```

.....

**PROGRAM 14**

**Write a C++ program to create a class called BIN\_TREE(Binary tree) with member functions to perform inorder, preorder and postorder traversals. Create a BIN\_TREE object and demonstrate the traversals.**

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

class node
{
public: int info;
        node *left;
        node *right;
};

typedef node *NODE;

/* declaring a class */
class BIN_TREE
{
public: node *root;

public: BIN_TREE()
        {
                root=NULL;
        }

/*function to get inorder expression*/
        void inorder(node *root)
        {
                if(root!=NULL)
                {
                        inorder(root->left);
                        cout<<root->info<<"\t";
                        inorder(root->right);
                }
        }

/*function to get preorder expression*/
        void preorder(node *root)
        {
                if(root!=NULL)
                {
                        cout<<root->info<<"\t";
                        preorder(root->left);
                        preorder(root->right);
                }
        }
};
```

```

        }
    }
}
/*function to get preorder expression*/
void postorder(node *root)
{
    if(root!=NULL)
    {
        postorder(root->left);
        postorder(root->right);
        cout<<root->info<<"\t";
    }
}
/*function to insert elements into binary tree*/
NODE insert(int item,node *root)
{
    node *temp,*prev,*cur;
    temp=new node;
    temp->info=item;
    temp->left=NULL;
    temp->right=NULL;
    if(root==NULL)
        return temp;
    prev=NULL;
    cur=root;
    while(cur!=NULL)
    {
        prev=cur;
        if(item==cur->info)
        {
            cout<<"Duplicate elements are not allowed"<<endl;
            delete(temp);
            return root;
        }
        cur=(item<cur->info)?cur->left:cur->right;
    }
    if(item<prev->info) prev->left=temp;
    else prev->right=temp;
    return root;
}
};

/* main function */
void main()
{
    BIN_TREE bin;
    int choice,item;

```

```
for(;;)
{
    cout<<"1.Insert 2.Inorder 3.Preorder 4.Postorder 5.Exit"<<endl;
    cout<<"Enter choice:";
    cin>>choice;
    switch(choice)
    {
        case 1: cout<<"Enter item:";
                cin>>item;
                bin.root=bin.insert(item,bin.root);
                break;

        case 2: if(bin.root==NULL)
                cout<<"Tree is empty"<<endl;
                else
                {
                    cout<<"Inorder Traversal:"<<endl;
                    bin.inorder(bin.root);
                    cout<<endl;
                }
                break;

        case 3: if(bin.root==NULL)
                cout<<"Tree is empty"<<endl;
                else
                {
                    cout<<"Preorder Traversal:"<<endl;
                    bin.preorder(bin.root);
                    cout<<endl;
                }
                break;

        case 4: if(bin.root==NULL)
                cout<<"Tree is empty"<<endl;
                else
                {
                    cout<<"Postorder Traversal:"<<endl;
                    bin.postorder(bin.root);
                    cout<<endl;
                }
                break;

        default: exit(0);
    }
}
}
```

**OUTPUT**  
.....

```
1.Insert 2.Inorder 3.Preorder 4.Postorder 5.Exit
Enter choice:1
Enter item:50
1.Insert 2.Inorder 3.Preorder 4.Postorder 5.Exit
Enter choice:1
Enter item:25
1.Insert 2.Inorder 3.Preorder 4.Postorder 5.Exit
Enter choice:1
Enter item:23
1.Insert 2.Inorder 3.Preorder 4.Postorder 5.Exit
Enter choice:1
Enter item:34
1.Insert 2.Inorder 3.Preorder 4.Postorder 5.Exit
Enter choice:1
Enter item:60
1.Insert 2.Inorder 3.Preorder 4.Postorder 5.Exit
Enter choice:1
Enter item:56
1.Insert 2.Inorder 3.Preorder 4.Postorder 5.Exit
Enter choice:1
Enter item:70
1.Insert 2.Inorder 3.Preorder 4.Postorder 5.Exit
Enter choice:2
Inorder Traversal:
23  25  34  50  56  60  70
1.Insert 2.Inorder 3.Preorder 4.Postorder 5.Exit
Enter choice:3
Preorder Traversal:
50  25  23  34  60  56  70
1.Insert 2.Inorder 3.Preorder 4.Postorder 5.Exit
Enter choice:4
Postorder Traversal:
23  34  25  56  70  60  50
1.Insert 2.Inorder 3.Preorder 4.Postorder 5.Exit
Enter choice:5
Press any key to continue
```

  
.....

**PROGRAM 15****Problem:**

**Write a C++ program to create a class called EXPRESSION. Using appropriate member functions convert a given valid infix expression to postfix form. Display the infix and postfix expressions.**

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

/* declaring a class */
class expression
{
    int top;
    char stack[20],infix[20],postfix[20];
    public: expression() { top=-1; }
        void readdata();
        void printdata();
        void infixtopostfix();
        void push(char ch);
        char pop();
        int preced(char ch);
};

/* function to read the infix expression*/
void expression::readdata()
{
    cout<<"\nEnter the infix expression:";
    cin>>infix;
}

/* function to print the expressions*/
void expression::printdata()
{
    cout<<"\nInfix expression is:"<<infix;
    cout<<"\nPostfix expression is:"<<postfix;
}
```

```
/* function to convert infix expression to postfix expression*/
```

```
void expression::infixtopostfix()
{
    int i,p;
    char ch;
    i=0;p=0;
    push('#');
    while(infix[i]!='\0')
    {
        ch=infix[i];
        switch(ch)
        {
            case '(':push(ch);
                break;
            case ')':while(stack[top]!='(')
                postfix[p++]=pop();
                pop();
                break;
            case '$':
            case '^':
            case '*':
            case '/':
            case '+':
            case '-':while(preced(stack[top])>=preced(ch))
                postfix[p++]=pop();
                push(ch);
                break;
            default:postfix[p++]=ch;
                break;
        }
        i++;
    }
    while(stack[top]!='#')
        postfix[p++]=pop();
    postfix[p]='\0';
}
```

```
/* function to push an element into the stack*/
```

```
void expression::push(char ch)
{
    stack[++top]=ch;
}
```

```
/* function to pop an element from the stack*/
```

```
char expression::pop()
```

```
{
    char ch=stack[top--];
    return ch;
}

/* function to check the precedence*/
int expression::preced(char ch)
{
    int p;
    switch(ch)
    {
        case '$':
        case '^':p=3;
            break;
        case '/':
        case '*':p=2;
            break;
        case '+':
        case '-':p=1;
            break;
        case '(':
        case ')':p=0;
            break;
        case '#':p=-1;
            break;
    }

    return p;
}

/* main function */
void main()
{
    expression e; clrscr();
    e.readdata();
    e.infixtopostfix();
    e.printdata();
    getch();
    cout<<"\nProgram Terminated";
    getch();
}
```

.....

**OUTPUT**

Enter the infix expression:((A\*B)\$C^F/E)

Infix expression is:((A\*B)\$C^F/E)

Postfix expression is:AB\*CF^E/

Program Terminated

.....