1. **Using circular representation for a polynomial, design, develop, and execute a program in C to accept two polynomials, add them, and then print the resulting polynomial.**

```
include <stdio.h>
#include <stdlib.h>
struct node
{   int coef;
   int expo;
   struct node *link;
};
struct node *attach(int coef1,int expo1,struct node *start)
{
   struct node *temp,*dstart;
   temp=(struct node *)malloc(sizeof(struct node));
   temp->coef=coef1;
   temp->expo=expo1;
   dstart=start->link;
   while(dstart->link!=start)
      dstart=dstart->link;  //the great inner meaning
   dstart->link=temp;
   temp->link=start;
   return start;
}
struct node *readpoly(struct node *start)
{
   char ch='y';
   int coef,expo;
   while(ch=='y')
   {
      printf("Enter Coefficient and Exponent\n");
      scanf("%d%d",&coef,&expo);
```

```
        start=attach(coef,expo,start);
        printf("Do you wish to enter any more enter(Y/N)\n");
        ch=getchar();
        scanf("%c",&ch);
    }
    return start;
}
int compare(int a,int b)
{
    if(a<b)
        return -1;
    if(a==b)
        return 0;
    if(a>b)
        return 1;
}
struct node *polyadd(struct node *startA,struct node *startB,struct node *startC)
{
    int sum;
    struct node *dstartA,*dstartB;
    dstartA=startA->link;
    dstartB=startB->link;
    while(dstartA!=startA && dstartB!=startB)
    {
        switch(compare(dstartA->expo,dstartB->expo))
        {
            case 1:
                startC=attach(dstartA->coef,dstartA->expo,startC);
                dstartA=dstartA->link;
                break;
            case 0:
```

```
                sum=dstartA->coef+dstartB->coef;
                startC=attach(sum,dstartA->expo,startC);
                dstartA=dstartA->link;
                dstartB=dstartB->link;
                break;
            case -1:
                startC=attach(dstartB->coef,dstartB->expo,startC);
                dstartB=dstartB->link;
                break;
        }
    }
    while(dstartA!=startA)
    {
        startC=attach(dstartA->coef,dstartA->expo,startC);
        dstartA=dstartA->link;
    }
    while(dstartB!=startB)
    {
        startC=attach(dstartB->coef,dstartB->expo,startC);
        dstartB=dstartB->link;
    }
    return startC;
}
int main()
{
    struct node *startA,*startB,*startC,*dstartA,*dstartB,*dstartC;

    startA=(struct node *)malloc(sizeof(struct node));
    startB=(struct node *)malloc(sizeof(struct node));
    startC=(struct node *)malloc(sizeof(struct node));
    startA->link=startA;// critical
```

```
startB->link=startB;
startC->link=startC;
printf("Reading Polynomial A\n");
startA=readpoly(startA);
printf("Reading Polynomial B\n");
startB=readpoly(startB);
dstartA=startA->link;
dstartB=startB->link;
printf("Polynomial A\n");
while(dstartA!=startA)
{
      printf("%d^%d",dstartA->coef,dstartA->expo);
   dstartA=dstartA->link; // critical
   if(dstartA!=startA)
      printf("+");
}
printf("\nPolynomial B\n");
while(dstartB!=startB)
{
      printf("%d^%d",dstartB->coef,dstartB->expo);
   dstartB=dstartB->link;
   if(dstartB!=startB)
      printf("+");
}
startC=polyadd(startA,startB,startC);
dstartC=startC->link;
printf("\nPolynomial C\n");
 while(dstartC!=startC)
{
      printf("%d^%d",dstartC->coef,dstartC->expo);
   dstartC=dstartC->link;
```

```
        if(dstartC!=startC)
            printf("+");
    }
    return 0;
}
```

2. **Design, develop, and execute a program in C to convert a given valid parenthesized infix arithmetic expression to postfix expression and then to print both the expressions. The expression consists of single character operands and the binary operators + (plus), -(minus), * (multiply) and / (divide).**

```
#include <stdio.h>
#include <stdlib.h>
typedef enum {lparen,rparen,plus,minus,times,divide,mod,eos,operand} precedence;
precedence token;
int stack[10];
int top=-1;
char expr[100];
precedence get_token(char*symbol,int*n)
{
   *symbol = expr[*n];
   *n=*n+1;
   switch (*symbol)
   {
      case '(' : return(lparen);break;
      case ')' : return(rparen);break;
      case '+' : return(plus);break;
      case '-' : return(minus);break;
      case '*' : return(times);break;
      case '/' : return(divide);break;
      case '%' : return(mod);break;
      case '\0' : return(eos);break;
      default : return(operand);break;
   }
}
```

```
void push (int a)
{
   if(top>=9)
   {
      printf("Stack overflow \n");
      return;
   }
   else
   {
      top=top+1;
      stack[top]=a;
   }
}
int pop()
{
   int b;
   if(top==-1)
   {
      printf("Stack is empty \n");
      return 0;
   }
   else
   {
      b=(stack[top]);
      top=top-1;
      return b;
   }
}
int eval()
{
   char symbol;
```

```
    int op1,op2;
    int n=0;
    token=get_token(&symbol,&n);
    while(token!=eos)
    {
       if(token==operand)
       {
          push (symbol-'0');
       }
       else
       {
          op2=pop();
          op1=pop();
          switch(token)
          {
             case plus : push(op1+op2);break;
             case minus : push(op1-op2);break;
             case times : push(op1*op2);break;
             case divide : push(op1/op2);break;
             case mod : push(op1%op2);break;
          }
         token=get_token(&symbol,&n);
    }
    return pop();
}
}
int main()
{
    int ans;
    printf("Enter the postfix expression \n");
    gets(expr);
```

```
    ans=eval();
    printf("The answer is %d",ans);
    return 0;
}
```

3. Design, develop, and execute a program in C to evaluate a valid postfix expression using stack. Assume that the postfix expression is read as a single line consisting of non-negative single digit operands and binary arithmetic operators. The arithmetic operators are + (add), - (subtract), * (multiply) and / (divide).

```
#include<iostream>
#include<cstdlib>
using namespace std;
class stack{
   public:
   int top,a[10];
   stack operator+(int ele);
   stack operator--(int x);
   friend ostream &operator<<(ostream &stream,stack s);
   stack(){top=-1;} //critical
};
stack stack:: operator+(int ele)
{
   if(top==9)
   {
      cout<<"Stack full"<<endl;
      return *this;
   }
   top++;
   a[top]=ele;
   return *this;
}
stack stack:: operator--(int x)
{
   if(top==-1)
   {
```

```
        cout<<"Stack empty"<<endl;
        return *this;
    }
    top--;
    return *this;
}
ostream &operator<<(ostream &stream,stack s)
{
    int i;
    if(s.top==-1)
    {
        stream<<"Stack empty"<<endl;
        return stream;
    }
    for(i=0;i<=s.top;i++) // for(i=s.top;i>=0;i--) for reverse display
    cout<<s.a[i];
    return stream;
}
int main()
{
   int ele; int ch; stack s1;
    cout<<"1-insert"<<endl<<"2-delete"<<endl<<"3-display"<<endl<<"4-exit"<<endl;

    for(;;)
    {
        cout<<"enter choice"<<endl;
        cin>>ch;
        switch(ch)
        {
            case 1: cout<<"Enter the item to be inserted"<<endl;
                    cin>>ele;
```

```
                s1=s1+ele;
                break;
        case 2: s1=s1--;
                break;
        case 3: cout<<s1; break;
        default : exit(0);
      }
   }
   return 0;
}
```

4. Design, develop, and execute a program in C to simulate the working of a queue of integers using an array. Provide the following operations:

a. Insert b. Delete c. Display

```
#include <stdio.h>
#define MAX 3
int front=-1;
int rear=-1;
int queue[MAX];
void insertq(int item)
{
   if(rear==MAX-1)
   {
      printf("queue is full\n");
      return;
   }
   rear+=1;
   queue[rear]=item;
   return ;
}
int deleteq()
{
   int item;
   if(rear==front)
   {
      printf("Queue is Empty\n");
      return;
   }
   front+=1;
   item=queue[front];
   return item;
```

```
}
void display()
{
   int i;
   if(rear==front)
   {
      printf("Queue is Empty\n");
      return;
   }
   for(i=front+1;i<=rear;i++)
   printf("%d",queue[i]);
   return ;
}
int main()
{
   int ch,item;
   printf("1-insert\n 2-delete \n 3- display 4-exit\n");
   for(;;)
  {
     printf("enter choice\n");
     scanf("%d",&ch);
   switch(ch)
   {
      case 1:printf("Enter the element to be inserted\n");
             scanf("%d",&item);
             insertq(item);
             break;
      case 2:item=deleteq();
              printf("item deleted is %d",item);
             break;
      case 3:display();
```

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

5. **Design, develop, and execute a program in C++ based on the following requirements:**

   An EMPLOYEE class is to contain the following data members and member functions:

   Data members: Employee_Number (an integer), Employee_Name (a string of characters), Basic_Salary (an integer), All_Allowances(an integer), IT (an integer), Net_Salary (an integer).

   Member functions: to read the data of an employee, to calculate Net_Salary and to print the values of all the data members. (All_Allowances = 123% of Basic; Income Tax (IT) = 30% of the gross salary (= basic_Salary _ All_Allowance); Net_Salary =Basic_Salary + All_Allowances – IT)

```
#include <iostream>
using namespace std;
class employee{
   public: char name[20];
        float basic,da,it,sal,netsal;
        int num;
        void getdata();
        void netsalary();
        void display();
};
void employee :: getdata()
{
   cout<<"enter name"<<endl; cin>>name;
```

```
    cout<<"Enter number"<<endl; cin>>num;
    cout<<"Enter basic"<<endl;  cin>>basic;
    return ;
}
void employee :: netsalary()
{
    da=0.52*basic;
    sal=da+basic;
    it=0.3*sal;
    netsal=sal-it;
    return;
}
void employee :: display()
{
cout<<name<<endl<<num<<endl<<basic<<endl<<da<<endl<<it<<endl<<netsal<<endl;
return ;
}

int main()
{
    int i,n;
```

```
    employee e[20];
    cout<<"Enter the no of employees"<<endl;
    cin>>n;
    for(i=0;i<n;i++)
    {
       cout<<"Enter the employee details"<<i+1<<endl;
       e[i].getdata();
       e[i].netsalary();
    }
    cout<<"NAME    BASIC    DA    IT    NETSAL"<<endl;
    for(i=0;i<n;i++)
    {
      e[i].display();
    }
    return 0;
}
```

6. Design, develop, and execute a program in C++ 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)

```
#include<iostream>
#include<cstring>
using namespace std;
class strings{
   char name[100];
   public:
   strings(){}
   strings(char *s)
   {
      strcpy(name,s);
   }
   strings operator+(strings s2)
   {
      strings s3;
      strcpy(s3.name,name);
```

```
        strcat(s3.name,s2.name);
        return s3;
    }
   friend ostream &operator<<(ostream &stream, strings s);
};
 ostream &operator<<(ostream &print, strings s)
 {
    print<<s.name;
    return print;
 }
 int main()
 {
    strings s1="VTU";
    strings s2="BELGAUM";
    strings s3=s1+s2;
    cout<<s3;
    return 0;
 }
#include<iostream>
#include<cmath>
using namespace std;
```

```
class octal{
    int o;
     public:
    octal(){}
    octal(int a){o=dectooct(a);}
    int dectooct(int a);
    int octtodec(int a);
    int operator+(int a)
    {
        return octtodec(o)+a;
    }
    friend ostream &operator<<(ostream &stream,octal ob);
};
int octal :: dectooct(int a)
{
    int r,s=0,i=0;
    while(a!=0)
    {
        r=a%8;
        s=s+r*pow(10,i);
        i++;
```

```
        a=a/8;
    }
    return s;
}
int octal :: octtodec(int b)
{
    int r,s=0,i=0;
    while(b!=0)
    {
        r=b%10;
        s=s+r*pow(8,i);
        b=b/10;
        i++;
    }
    return s;
}
ostream &operator<<(ostream &stream,octal ob)
{
    cout<<ob.o<<endl;
    return stream;
}
```

```
int main()
{
   int val1=100;
   int val2 =200;
   octal ob=val1;
   int y;
   y=ob+val2;
   cout<<ob;
   cout<<y;
   return 0;
}
```

7. **Design, develop, and execute a program in C++ to create a class called STACK using an array of integers and to 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 to top of the stack.

   ii. s1=s1- ; where s1 is an object of the class STACK and – operator pops off the top 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>
#include<cstdlib>
using namespace std;
class stack{
   public:
   int top,a[10];
   stack operator+(int ele);
   stack operator--(int x);
   friend ostream &operator<<(ostream &stream,stack s);
   stack(){top=-1;} //critical
};
stack stack:: operator+(int ele)
{
```

```
    if(top==9)
    {
        cout<<"Stack full"<<endl;
        return *this;
    }
    top++;
    a[top]=ele;
    return *this;
}
stack stack:: operator--(int x)
{
    if(top==-1)
    {
        cout<<"Stack empty"<<endl;
        return *this;
    }
    top--;
    return *this;
}
ostream &operator<<(ostream &stream,stack s)
{
```

```
    int i;
    if(s.top==-1)
    {
        stream<<"Stack empty"<<endl;
        return stream;
    }
    for(i=0;i<=s.top;i++)
    cout<<s.a[i];
    return stream;
}
int main()
{
   int ele; int ch; stack s1;
    cout<<"1-insert"<<endl<<"2-delete"<<endl<<"3-display"<<endl<<"4-exit"<<endl;
    for(;;)
    {
        cout<<"enter choice"<<endl;
        cin>>ch;
        switch(ch)
        {
            case 1: cout<<"Enter the item to be inserted"<<endl;
```

```
            cin>>ele;
            s1=s1+ele;
            break;
        case 2: s1=s1--;
            break;
        case 3: cout<<s1; break;
        default : exit(0);
      }
   }
   return 0;
}
```

8. Design, develop, and execute a program in C++ 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.

```
#include<iostream>
#include<cstdlib>
using namespace std;
class node{
   public:
   node *link;
   int data;
};
class list{
   public:
   node *start;
   void insert(int item);
   int del();
   void display();
   list(){start=NULL;}
   ~list(){}
};
void list :: insert(int item)
{
   node *temp=new node; //critical
   temp->data=item;
   temp->link=NULL;
   if(start!=NULL)
   {
      temp->link=start;
   }
```

```
    start=temp;
    return ;
}
int list :: del()
{   int x;
    node *dstart;
    if(start==NULL)
    {
        cout<<"List empty"<<endl;
         return NULL;
    }
    dstart=start;
    start=start->link;
    x=dstart->data;
    free(dstart);
    return x;
}
void list :: display()
{
     if(start==NULL)
    {
        cout<<"List empty"<<endl;
        return ;
    }
    node *dstart;
    dstart=start;
    while(dstart!=NULL)
          {
             cout<<dstart->data;
             dstart=dstart->link;
          }
```

```
        return ;
}
int main()
{
   list L;int ch,item;
   cout<<"1-insert"<<endl<<"2-delete"<<endl<<"3-display"<<endl<<"4-exit"<<endl;
    for(;;)
   {
      cout<<"enter choice"<<endl;
      cin>>ch;
      switch(ch)
      {
         case 1:  cout<<"enter the element to be inserted"<<endl;
         cin>>item;
                L.insert(item);
                break;
         case 2:item=L.del();
               cout<<item;
               break;
         case 3: L.display();
                break;
         default: exit(0);
      }
   }
   return 0;
}
```

9. Design, develop, and execute a program in C to read a sparse matrix of integer values and to search the sparse matrix for an element specified by the user. Print the result of the search appropriately. Use the triple <row, column, value> to represent an element in the sparse matrix.

```
#include<stdio.h>
struct smatrix{
   int row,col,val;
}s[10],b[10];
int main()
{
   int m,n,i,j,lvalue=0,l=1,a[10][10],key,flag=0,k=1;
   printf("Enter the no of rows and columns\n");
   scanf("%d %d",&m,&n);
   printf("Enter the elements of the matrix\n");
   for(i=0;i<m;i++)
   for(j=0;j<n;j++)
   scanf("%d",&a[i][j]);
   for(i=0;i<m;i++)
   for(j=0;j<n;j++)
   {
      if(a[i][j]!=0)
      {
         s[l].row=i;
         s[l].col=j;
         s[l].val=a[i][j];
         l++;
         lvalue++;
    }   }
      s[0].row=i;
      s[0].col=j;
```

```
   s[0].val=lvalue;
printf("ROW COL VALUE\n");
for(i=0;i<=s[0].val;i++)
{
   printf("%d   %d    %d\n",s[i].row,s[i].col,s[i].val);
}
printf("Enter the key to be searched\n");
scanf("%d",&key);
for(i=1;i<=s[0].val;i++)
{
   if(key==s[i].val)
   {
      flag=1;
      break;
   }
}
if(flag==1) printf("element %d is found in %d row and %d column",key,s[i].row,s[i].col);
else printf("not found\n");
k=1;
for(i=0;i<s[0].col;i++)
{
   for(j=1;j<=s[0].val;j++)
   {
      if(i==s[j].col)
      {
         b[k].row=s[j].col;
         b[k].col=s[j].row;
         b[k].val=s[j].val;
         k++;
      }
   }
```

```
    }
    b[0].row=s[0].col;
            b[0].col=s[0].row;
            b[0].val=s[0].val;
     printf("ROW COL VALUE\n");
    for(i=0;i<=b[0].val;i++)
    {
       printf("%d   %d    %d\n",b[i].row,b[i].col,b[i].val);
    }
    return 0;
}
```

10. Design, develop, and execute a program in C to create a max heap of integers by accepting one element at a time and by inserting it immediately in to the heap. Use the array representation for the heap. Display the array at the end of insertion phase.

```
#include <stdio.h>
#include <stdlib.h>
# define MAX 100
int h[MAX];
int n=0;
void insert(int element)
{
   int childpo,parentpo;
   childpo=n;
   parentpo=(childpo-1)/2;
   h[childpo]=element;
   while(childpo!=0 && h[childpo]>h[parentpo])
   {
      h[childpo]=h[parentpo];
      h[parentpo]=element;
      childpo=parentpo;
      parentpo=(childpo-1)/2;
   }
   h[childpo]=element;
   n=n+1;
}
void deleteh()
{
   int parentpo,childpo;
   printf("Element deleted is %d\n",h[0]);
   parentpo=0;
   childpo=parentpo*2+1;
```

```
    while(childpo<n)
    {
        if(h[childpo]<h[childpo+1] && (childpo+1)<n)
            childpo=childpo+1;
        h[parentpo]=h[childpo];
        parentpo=childpo;
        childpo=parentpo*2+1;
    }
    n=n-1;
}
void display()
{
    int i;
    printf("Elements of the heap are:\n");
    for(i=0;i<n;i++)
    {
 printf("%d\n",h[i]);
    }
}
int main()
{
    int ch,element;
    while(1)
    {
        printf("Enter your choice\n");
        printf("1.Insert  2.Delete  3.Display 4-exit\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
                printf("Enter the element to be inserted\n");
```

```
                scanf("%d",&element);
                insert(element);
                break;
        case 2:
                deleteh();
                break;
        case 3:
                display();
                break;
        default:
                exit(0);
      }
   }
   return 0;
}
```

11. Design, develop, and execute a program in C to implement a doubly linked list where each node consists of integers. The program should support the following operations:

i. Create a doubly linked list by adding each node at the front.

ii. Insert a new node to the left of the node whose key value is read as an input.

iii. Delete the node of a given data if it is found, otherwise display appropriate message.

iv. Display the contents of the list.

(Note: Only either (a,b and d) or (a, c and d) may be asked in the examination)

```
#include <stdio.h>
#include <stdlib.h>
struct node{
   int data;
   struct node *llink;
   struct node *rlink;
}*start;
void insert(int item)
{
   struct node *temp;
   temp=(struct node*)malloc(sizeof(struct node));
   temp->data=item;
   temp->llink=temp->rlink=NULL;
```

```
    if(start!=NULL)
    {
        temp->rlink=start;
        start->llink=temp;
    }
    start=temp;
    return;
}
void insertbfx(int item)
{
    struct node *temp,*dstart;
    int value;
    temp=(struct node*)malloc(sizeof(struct node));
    printf("Enter the data to be inserted\n");
    scanf("%d",&value);
    temp->data=value;
    dstart=start;
    while(dstart!=NULL)
    {
        if(dstart->data==item)
        {
```

```
        if(dstart==start)
        {
           temp->rlink=start;
           start->llink=temp; start=temp;//critical
           return ;
        }
        else{
           temp->rlink=dstart;
           temp->llink=dstart->llink;
           (dstart->llink)->rlink=temp;
           dstart->llink=temp;
           return ;
        }
      }
      dstart=dstart->rlink;
   }
  printf("Not found\n");
   return ;

}
void deletex(int item)
```

```
{
   struct node* dstart;
   dstart=start;
   if(start==NULL)
   {
      printf("DLL empty\n");
      return ;
   }
   while(dstart!=NULL)
   {
      if(dstart->data==item)
      {
         if(dstart==start)
         {
            if(start->rlink==NULL)
            {
               printf("Item deleted\n");
                  free(start); start=NULL;//critical
                   return;
            }
            start=start->rlink;
```

```
            start->llink=NULL;
            free(dstart); //critical
            return ;
          }
          else{
            if(dstart->rlink==NULL){printf("Item deleted\n"); (dstart->llink)->rlink=NULL;//critical
             free(dstart); return;}
            (dstart->llink)->rlink=dstart->rlink;
            (dstart->rlink)->llink=dstart->llink;
            printf("Item deleted\n"); free(dstart); return;
          }
       }
       dstart=dstart->rlink;
    }
    printf("element not found\n");
   return ;
}
void display()
{
    struct node *dstart;
```

```
    dstart=start;
     if(start==NULL)
    {
       printf("DLL empty\n");
       return ;
    }
    while(dstart!=NULL)
    {
       printf("%d",dstart->data);
       dstart=dstart->rlink;
    }
    return ;
}
int main()
{
    int ch,item;
    printf("1-insert\n 2-insertbefore x \n 3- deletex  4-display\n 5-exit\n");
    for(;;)
   {
       printf("enter choice\n");
       scanf("%d",&ch);
```

```
  switch(ch)
  {
     case 1:printf("Enter the element to be inserted\n");
            scanf("%d",&item);
            insert(item);
            break;
     case 2: printf("Enter the element x before which data to be added\n");
            scanf("%d",&item);
            insertbfx(item);
            break;
     case 3: printf("Enter the element to be deleted\n");
            scanf("%d",&item);
            deletex(item);
            break;
     case 4: display(); break;
     default: exit(0);
  }
 }
 return 0;
}
```

12. Design, develop, and execute a program in C++ to create a class called DATE with methods to accept two valid dates in the form dd/mm/yy and to implement the following operations by overloading the operators + and -. After every operation the results are to be displayed 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>
using namespace std;
int leapm[]={0,31,29,31,30,31,30,31,31,30,31,30,31};
int nleapm[]={0,31,28,31,30,31,30,31,31,30,31,30,31};
class DATE{
   public:
   int dd,mm,yy,leapyear;
   void getdate();
   friend int operator-(DATE d1, DATE d2);
   DATE operator+(int nod);
   friend ostream &operator<<(ostream &stream,DATE obj);
};
void DATE :: getdate()
{
```

```
    int valid=0;
    leapyear=0;
    do{
        cout<<"Enter year"<<endl;
        cin>>yy;
        if(yy%4==0)
        leapyear=1;
        cout<<"Enter month"<<endl;
        cin>>mm;
        if(mm<13)
        valid=1;
        cout<<"Enter day"<<endl;
        cin>>dd;
        if(leapyear==1 && dd<=leapm[mm] || leapyear==0 && dd<=nleapm[mm])
        valid=1;
    }while(valid==0);
    return ;
}
int operator-(DATE d1,DATE d2)
{
    int count=0;
```

```
    while(d1.mm!=d2.mm || d1.yy!=d2.yy)
    {
        if(d2.leapyear==1)
        count=count+leapm[d2.mm];
        else count=count+nleapm[d2.mm];
        d2.mm++;
        if(d2.mm>12)
        {
            d2.mm=1;
            d2.yy++;
            if(d2.yy%4==0)
            d2.leapyear=1;
        }
    }
    count=count+d1.dd-d2.dd;
    return count;
}
DATE DATE :: operator+(int nod)
{
    nod=nod+dd;
    dd=0;
```

```
    while(leapyear==1 && nod>leapm[mm] || leapyear ==0 && nod>nleapm[mm])
    {
        if(leapyear==1)
        nod=nod-leapm[mm];
        else
        nod=nod-nleapm[mm];
        mm++;
        if(mm>12)
        {
            mm==1;
            yy++;
            if(yy%4==0)
            leapyear=1;
        }
    }
    dd=nod;
    return *this;
}
ostream &operator<<(ostream &stream,DATE obj)
{
    cout<<obj.dd<<"/"<<obj.mm<<"/"<<obj.yy<<endl;
```

```
    return stream;
}
int main()
{
    DATE d1,d2; int nod;
    cout<<"enter date 1"<<endl;
    d1.getdate();
    cout<<"enter date 2"<<endl;
    d2.getdate();
    cout<<d1-d2;
    cout<<"Enter the value to be added to date 1"<<endl;
    cin>>nod;
    cout<<d1+nod;
    return 0;
}
#include<iostream>
using namespace std;
struct bnode{
    int data;
    struct bnode *left;
    struct bnode *right;
```

```
};
class bin_tree{
 public: struct bnode *root;
    bin_tree(){root=NULL;}
    ~bin_tree(){ }
    void inorder(struct bnode*root);
    void preorder(struct bnode*root);
    void postorder(struct bnode*root);
    struct bnode *make_tree(struct bnode *root,int ele);
};
void bin_tree :: inorder(struct bnode *root)
{
    if(root!=NULL)
    {
        inorder(root->left);
        cout<<root->data;
        inorder(root->right);
    }
    return ;
}
void bin_tree :: preorder(struct bnode *root)
```

```
{
   if(root!=NULL)
   {
      cout<<root->data;
      preorder(root->left);
      preorder(root->right);
   }
   return ;
}
void bin_tree :: postorder(struct bnode *root)
{
   if(root!=NULL)
   {
      postorder(root->left);
      postorder(root->right);
      cout<<root->data;
   }
   return ;
}
struct bnode * bin_tree :: make_tree(struct bnode *root,int ele)
{
```

```
    if(root==NULL)
    {
        struct bnode *root=new struct bnode;
        root->data=ele;
        root->left=root->right=NULL;
        cout<<"Node"<<ele<<"attached"<<endl;
        return root;
    }
    else if(ele<=root->data)
    root->left=make_tree(root->left,ele);
    else
    root->right=make_tree(root->right,ele);
    return root;
}
int main()
{
    int i,n,ele;
    bin_tree B; struct bnode *root=NULL;
    cout<<"Enter the no. of elements"<<endl;
    cin>>n;
    for(i=0;i<n;i++)
```

```
    {
       cout<<"Enter the element"<<endl;
       cin>>ele;
       root=B.make_tree(root,ele);
    }
    cout<<"Preorder traversal"<<endl;
    B.preorder(root);
     cout<<"inorder traversal"<<endl;
    B.inorder(root);
     cout<<"Postorder traversal"<<endl;
    B.postorder(root);
    return 0;
}
```

13. Design, develop, and execute a program in C++ 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. OCTAL h = x ; where x is an integer

ii. 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.

```
#include<iostream>
#include<cmath>
using namespace std;
class octal{
   int o;
    public:
   octal(){}
   octal(int a){o=dectooct(a);}
   int dectooct(int a);
   int octtodec(int a);
   int operator+(int a)
   {
      return octtodec(o)+a;
   }
```

```
    friend ostream &operator<<(ostream &stream,octal ob);
};
int octal :: dectooct(int a)
{
    int r,s=0,i=0;
    while(a!=0)
    {
        r=a%8;
        s=s+r*pow(10,i);
        i++;
        a=a/8;
    }
    return s;
}
int octal :: octtodec(int b)
{
    int r,s=0,i=0;
    while(b!=0)
    {
            r=b%10;
            s=s+r*pow(8,i);
```

```
        b=b/10;
        i++;
    }
    return s;
}
ostream &operator<<(ostream &stream,octal ob)
{
    cout<<ob.o<<endl;
    return stream;
}
int main()
{
    int val1=100;
    int val2 =200;
    octal ob=val1;
    int y;
    y=ob+val2;
    cout<<ob;
    cout<<y;
    return 0;
}
```

14. Design, develop, and execute a program in C++ to create a class called BIN_TREE that represents a Binary Tree, with member functions to perform inorder, preorder and postorder traversals. Create a BIN_TREE object and demonstrate the traversals.

```
#include<iostream>
using namespace std;
class bnode{
public:    int data;
   bnode *llink;
   bnode *rlink;
};
class bintree{
  public:   bnode *root;
   bintree(){root=NULL;}
   ~bintree(){}
   bnode* make_tree(bnode* root,int element);
   void preorder(bnode *root);
   void postorder(bnode *root);
   void inorder(bnode *root);
};
bnode* bintree :: make_tree(bnode* root,int element)
{
   if(root==NULL)
   {
      bnode* root=new bnode; //critical
      root->data=element;
      root->llink=root->rlink=NULL;
      cout<<"Node"<<element<<"Attached"<<endl;
      return root;
   }
   if(element<=root->data)
```

```
    {
       root->llink=make_tree(root->llink,element);
    }
    else{
        root->rlink=make_tree(root->rlink,element);
    }
    return root;
}
void bintree :: preorder(bnode *root)
{   if(root!=NULL){
    cout<<root->data;
    preorder(root->llink);
    preorder(root->rlink);}
    return ;
}
void bintree :: postorder(bnode *root)
{
if(root!=NULL){
    postorder(root->llink);
    postorder(root->rlink);
    cout<<root->data;}
    return ;
}
void bintree ::inorder(bnode *root)
{
  if(root!=NULL)
  {
    inorder(root->llink);
    cout<<root->data;
    inorder(root->rlink);
    return ;
```

```
  }
}
int main()
{
   int n,ele,i; bintree B; bnode* root=NULL;  //critical
   cout<<"enter no of elements to be inserted"<<endl;
   cin>>n;
   for(i=0;i<n;i++)
   {
      cout<<"Enter the element"<<endl;
      cin>>ele;
      root=B.make_tree(root,ele);
   }
  cout<<"Inorder Traversal"<<endl;
  B.inorder(root);
  cout<<"postorder Traversal"<<endl;
  B.postorder(root);
  cout<<"preorder Traversal"<<endl;
  B.preorder(root);
  return 0;
}
/*
*/
#include<iostream>
#include<cstring>
using namespace std;
class strings{
   char name[100];
   public:
   strings(){}
   strings(char *s)
```

```
  {
     strcpy(name,s);
  }
  strings operator+(strings s2)
  {
     strings s3;
     strcpy(s3.name,name);
     strcat(s3.name,s2.name);
     return s3;
  }
  friend ostream &operator<<(ostream &stream, strings s);
};
ostream &operator<<(ostream &print, strings s)
{
   print<<s.name;
   return print;
}
int main()
{
   strings s1="VTU";
   strings s2="BELGAUM";
   strings s3=s1+s2;
   cout<<s3;
   return 0;
}
```