

**CONCORD ARTS AND SCIENCE COLLEGE**

**CONCORD EDUCITY, MUTTANNUR, (PO) PATTANNUR**



**FOURTH SEMESTER BACHELOR OF  
COMPUTER APPLICATION**

**PRACTICAL RECORD**

**2021-2022**

**DATA STRUCTURES  
AND DATABASE MANAGEMENT SYSTEM**

# **CONCORD ARTS AND SCIENCE COLLEGE**

**CONCORD EDUCITY, MUTTANNUR, (PO) PATTANNUR**



## **CERTIFICATE**

It is certified that this is a bonafide record of the original work done by  
Mr./Mrs..... Reg.no.....  
of IV<sup>th</sup> semester BCA in the data structures and database management  
system lab during the year 2021-2022.

**HOD:**

**Lecturer in charge:**

Submitted for practical examination held on .....

**External Examiner**

1.

2.

# INDEX

## DATA STRUCTURE

No.	PROGRAM	PAGE NO
1.	Add two Polynomials	
2.	Sequential and Binary search	
3.	Insertion Sort	
4.	Bubble Sort	
5.	Selection Sort	
6.	Quick Sort	
7.	Stack operation	
8.	Queue operation using array	
9.	Conversion of Infix expression to Postfix	
10.	operations on a Circular Queue	
11.	Singly Linked List	
12.	Circular Linked list	
13.	Doubly linked list	
14.	Implement tree traversal	
15	Merge to sort linked list	

# INDEX

## DATABASE MANAGEMENT SYSTEM

No.	PROGRAM	PAGE NO
1	SQL 1- Students Table	
2	SQL 2- Department Table 1	
3	SQL 3- Department Table 2	
4	SQL 4- Emp Table	
5	SQL 5- Department Table 3	
6	SQL 6- Customer Table	

# DATA STRUCTURE

1.

### AIM

Add two Polynomials

### PROGRAM

```
#include<iostream>
using namespace std;
class poly
{
    public:
    int poly1[20],poly2[20],poly3[20];
    int i,deg1,deg2,deg3;
    void read( );
};
void poly::read( )
{
    cout<<"enter degree of 1st poly:";
    cin>>deg1;
    cout<<"enter degree of 2nd poly:";
    cin>>deg2;
    cout<<"\n";
    cout<<"for first polynomiya:\n";
    for(i=0;i<=deg1;i++)
    {
        cout<<"enter coefficient of"<<i<<"::";
        cin>>poly1[i];
    }
}
```

```

cout<<"\n";
cout<<"for second polynomiyal:\n";
for(i=0;i<=deg2;i++)
{
    cout<<"enter coefficient of"<<i<<"::";
    cin>>poly2[i];
}
cout<<"\n polynomial 1:\n";
for(i=deg1;i>=0;i--)
{
    cout<<poly1[i]<<"x^"<<i;
    if(i>0)
    {
        cout<<"+";
    }
}
cout<<"\n";
cout<<"\n polynomial 2:\n";
for(i=deg2;i>=0;i--)
{
    cout<<poly2[i]<<"x^"<<i;
    if(i>0)
    {
        cout<<"+";
    }
}
deg3=(deg1>deg2)?deg1:deg2;

```

```

        for(i=0;i<=deg3;i++)
        {
            poly3[i]=poly1[i]+poly2[i];
        }
        cout<<"\n";
        cout<<"\nPolynomial after addition:\n";
        for(i=deg3;i>=0;i--)
        {
            cout<<poly3[i]<<"x^"<<i;
            if(i>0)
            {
                cout<<"+";
            }
        }
    }
}

int main( )
{
    poly po;
    po.read( );
    return 0;
}

```

## **OUTPUT**

Enter degree of 1st poly:3

Enter degree of 2nd poly:3



For first polynomiya:

Enter coefficient of 0 : 4

Enter coefficient of 1 : 3

Enter coefficient of 2 : 5

Enter coefficient of 3 : 10

For second polynomiya:

Enter coefficient of 0 : 6

Enter coefficient of 1 : 7

Enter coefficient of 2 : 8

Enter coefficient of 3 : 20

Polynomial 1:

$$10x^3+5x^2+3x^1+4x^0$$

Polynomial 2:

$$20x^3+8x^2+7x^1+6x^0$$

Polynomial after addition:

$$30x^3+13x^2+10x^1+10x^0$$

2.

### **AIM**

Sequential and Binary search

### **PROGRAM**

### **PROGRAM**

```
#include<iostream>

using namespace std;

static int flag=0;

class search
{
    int a[50],beg,end,loc,mid,i,n,item;

    public:

    void linearsearch( );

    void binarysearch();

};

void search::linearsearch()

{

    cout<<"Enter the limit";

    cin>>n;

    cout<<"Enter the elements of the array";

    for(i=0;i<n;i++)
```

```

{
    cin>>a[i];
}

cout<<"Enter the item to be searched";

cin>>item;

for(i=0;i<n;i++)
{
    if(a[i]==item)
    {
        flag=1;

        loc=i;
    }
}

if(flag==1)
{
    cout<<"\nSearch is successful";

    cout<<item<<" is found at location"<<loc+1;

}

else

{

```

```

        cout<<"search is unsuccessful";

    }

}

void search::binarysearch( )
{

    cout<<"Enter the limit";

    cin>>n;

    cout<<"Enter the elements of the array";

    for(i=0;i<n;i++)

    {

        cin>>a[i];

    }

    cout<<"Enter the item to be searched";

    cin>>item;

    beg=0;

    end=n-1;

    while(beg<=end)

    {

        mid=(beg+end)/2;

        if(item==a[mid])

```

```

        {

            cout<<"\nSearch is successful";

            loc=mid;

            cout<<"\n"<<item<<"is found at position "<<loc+1;

            flag=1;

            break;

        }

        else if(item<a[mid])

        {

            end=mid-1;

        }

        else

        {

            beg=mid+1;

        }

    }

    if(flag==0)

    {

        cout<<"\nSearch is unsuccessful";

    }

```

```

    }

int main( )
{
    search sr;

    int ch;

    cout<<"Menu 1. Linear search 2. Binary search ...Enter your choice:";

    cin>>ch;

    switch(ch)
    {

        case 1:

            sr.linearsearch();

            break;

        case 2:

            sr.binarysearch();

            break;

        default:

            cout<<"Invalid entry";

    }

    return 0;

}

```

## **OUTPUT**

Menu 1. Linear search 2. Binary search ...Enter your choice: 1

Enter the limit 4

Enter the elements of the array 56 78 23 1

Enter the item to be searched 23

Search is successful

23 is found at position 3

3.

### **AIM**

Insertion Sort

### **PROGRAM**

```
#include<iostream>

using namespace std;

class ins
{
    int i,j,k,temp,n,a[10];

    public:

    void sort( );

};

void ins::sort( )
{
    cout<<"Enter the limit";

    cin>>n;

    cout<<"Enter elements";

    for(i=0;i<n;i++)
    {

        cin>>a[i];
```



```

        }

for(k=0;k<n;k++)
{
    j=k-1;

    temp=a[k];

    while((temp<(a[j])&&(j>=0))
    {
        a[j+1]=a[j];
        j=j-1;
    }

    a[j+1]=temp;

}

cout<<"New array after sorting in ascending order";

for(i=0;i<n;i++)
{
    cout<<a[i]<<"\n";
}

}

int main( )
{

```

```
    ins in;  
  
    in.sort();  
  
    return 0;  
  
}
```

### **OUTPUT**

Enter the limit : 5

Enter the elements :    23    45    76    5    2

New array after sorting in ascending order :    2    5    23    45    76

4.

### **AIM**

Bubble Sort

### **PROGRAM**

```
#include<iostream>

using namespace std;

class bubble

{

    int a[100],i,j,n,temp;

    public:

    void sort( );

};

void bubble::sort( )

{

    cout<<"Enter the limit";

    cin>>n;

    cout<<"Enter the elements";

    for(i=0;i<n;i++)

    {

        cin>>a[i];
```

```

    }

    for(i=0;i<n;i++)

    {

        for(j=0;j<n-1;j++)

        {

            if(a[j]>a[j+1])

            {

                temp=a[j];

                a[j]=a[j+1];

                a[j+1]=temp;

            }

        }

    }

    cout<<"In Ascending order\n";

    for(i=0;i<n;i++)

    {

        cout<<a[i]<<"\t";

    }

}

```

```
int main( )  
  
{  
  
    bubble b;  
  
    b.sort( );  
  
    return 0;  
  
}
```

### **OUTPUT**

Enter the limit

5

Enter the elements

21 30 2 24 5

In ascending order

2 5 21 24 30

5.

**AIM**

Selection Sort

**PROGRAM**

```
#include<iostream>

using namespace std;

class sel

{

    int a[20],i,j,temp,min,n,loc;

    public:

    void sort( );

};

void sel::sort()

{

    cout<<"enter the limit";

    cin>>n;

    cout<<"enter elments";

    for(i=0;i<n;i++)

    {

        cin>>a[i];
```

```
}  
  
for(i=0;i<n;i++)  
{  
  
    min=a[i];  
  
    loc=i;  
  
    for(j=i+1;j<n;j++)  
    {  
  
        if(a[j]<min)  
        {  
  
            min=a[j];  
  
            loc=j;  
  
        }  
  
    }  
  
    if(loc!=i)  
    {  
  
        temp=a[i];  
  
        a[i]=a[loc];  
  
        a[loc]=temp;  
  
    }  
  
}
```

```

        cout<<"Array after Sorting\n";

        for(i=0;i<n;i++)

        {

                cout<<a[i]<<"\t";

        }

}

int main( )

{

        sel s;

        s.sort( );

        return 0;

}

```

## **OUTPUT**

Enter the limit

5

Enter the Elements

23

4



56

7

1

Array after Sorting

1      4      7      23      56

6.

**AIM**

Quick Sort

**PROGRAM**

```
#include<iostream>

using namespace std;

class quicksort

{

    public:

    int a[20],i,size,j,temp,pivot;

    void quick(int a[10] ,int first,int last);

};

int main( )

{

    int size,i,a[i];

    quicksort qs;

    cout<<"enter size";

    cin>>size;

    cout<<"enter elements";
```

```

        for(i=0;i<size;i++)
        {
                cin>>a[i];
        }

        qs.quick(a,0,size-1);

        cout<<"sorted array is";

        for(i=0;i<size;i++)
        {
                cout<<a[i]<<"\t";
        }

return 0;

}

void quicksort::quick(int a[10],int first,int last)
{
        if(first<last)
        {
                pivot=first;

                i=first;

                j=last;

```

```

while(i<j)
{
    while(a[i]<=a[pivot]&& i<last)
    {
        i++;
    }
    while(a[j]>a[pivot])
    {
        j--;
    }
    if(i<j)
    {
        temp=a[i];
        a[i]=a[j];
        a[j]=temp;
    }
}

temp=a[pivot];
a[pivot]=a[j];
a[j]=temp;

```

```
        quick(a,first,j-1);

        quick(a,j+1,last);

    }

}
```

### **OUTPUT**

Enter size:

5

Enter elements :

34

5

2

9

6

Sorted array is :

2      5      6      9      34

7.

### **AIM**

Stack operation

### **PROGRAM**

```
#include<iostream>

using namespace std;

#define maxsize 10

static int top=-1;

class stackop
{
    int a[maxsize],value,i;

    public:

        void push(int);

        void pop();

        void display();

};

void stackop::push(int item)
{
    if(top==maxsize-1)
    {
        cout<<"stack is full";
    }
    else
    {
```

```

        top=top+1;
        a[top]=item;
        cout<<"element is inserted";
    }
}

void stackop::pop()
{
    if(top==-1)
    {
        cout<<"stack is empty";
    }
    else
    {
        value=a[top];
        top=top-1;
        cout<<"poped element is:"<<value;
    }
}

void stackop::display()
{
    if(top==-1)
    {
        cout<<"stack is empty";
    }
}

```

```

else
{
    for(i=0;i<=top;i++)
    {
        cout<<a[i]<<"\n";
    }
}
}

int main()
{
    stackop stk;
    int item,ch;
    char c;
    do
    {
        cout<<"enter your choice:";
        cout<<"Menu 1.push 2.pop 3.display:";
        cin>>ch;
        switch(ch)
        {
            case 1:
                cout<<"enter item to be inserted:";
                cin>>item;
                stk.push(item);

```



```

        break;
    case 2:
        stk.pop();
        break;
    case 3:
        stk.display();
        break;
    default:
        cout<<"Invalid entry";

}
cout<<"Do you want to continue press y or n:";
cin>>c;
} while(c=='y' || c=='Y');
return 0;
}

```

## **OUTPUT**

enter your choice:Menu 1.push 2.pop 3.display:1

enter item to be inserted:10

element is insertedDo you want to continue press y or n:y

enter your choice:Menu 1.push 2.pop 3.display:1

enter item to be inserted:20

element is insertedDo you want to continue press y or n:y

enter your choice:Menu 1.push 2.pop 3.display:1

enter item to be inserted:30

element is insertedDo you want to continue press y or n:y

enter your choice:Menu 1.push 2.pop 3.display:3

10

20

30

Do you want to continue press y or n:y

enter your choice:Menu 1.push 2.pop 3.display:2

poped element is:30Do you want to continue press y or n:y

enter your choice:Menu 1.push 2.pop 3.display:3

10

20

Do you want to continue press y or n:y

8.

**AIM**

Queue operation using array

**PROGRAM**

```
#include<iostream>

#define maxsize 10

using namespace std;

class queueop
{
    int front , rear , Q[maxsize];
public:
    void insert();
    void deletion();
    void display();
    queueop()
    {
        front=-1;
        rear=-1;
    }
};

void queueop::insert()
{
    int num;

    if(rear==maxsize -1)
```

```

        {
            cout<<"Queue is full";
        }
    else
    {
        cout<<"enter the number to be inserted:";
        cin>>num;
        rear=rear+1;
        Q[rear]=num;
        if(front==-1)
        {
            front=0;
        }
    }
}

void queueop::deletion()
{
    int num;
    if(front==-1)
    {
        cout<<"Queue is empty";
    }
    else if(front==rear)
    {
        front=-1;
    }
}

```

```

        rear=-1;
    }
else
{
    num=Q[front];
    cout<<"deleted element is:"<<num;
    front=front+1;
}
}
void queueop::display()
{
    int i ;
    if(front==-1)
    {
        cout<<"queue is empty";
    }
    else
    {
        for(i=front;i<=rear;i++)
        {
            cout<Q[i]<<"\t";
        }
    }
}

```

```

int main()
{
    stackop stk;
    int item,ch;
    char c;
    do
    {
        cout<<"enter your choice:";
        cout<<"Menu 1.insertion  2. Deletion  3.display:";
        cin>>ch;
        switch(ch)
        {
            case 1:
                obj.insert();
                break;
            case 2:
                obj.deletion();
                break;
            case 3:
                obj.display();
                break;
            default:
                cout<<"Invalid entry";
        }
    }
}

```

```

cout << "Do you want to continue press y or n:";
cin >> c;
} while (c == 'y' || c == 'Y');
return 0;
}

```

## **OUTPUT**

```

enter your choice:Menu 1.insertion 2. Deletion 3.display:1
enter the number to be inserted:10
Do you want to continue press y or n:y
enter your choice:Menu 1.insertion 2. Deletion 3.display:1
enter the number to be inserted:20
Do you want to continue press y or n:y
enter your choice:Menu 1.insertion 2. Deletion 3.display:1
enter the number to be inserted:30
Do you want to continue press y or n:y
enter your choice:Menu 1.insertion 2. Deletion 3.display:3
10 20 30 Do you want to continue press y or n:y
enter your choice:Menu 1.insertion 2. Deletion 3.display:2
deleted element is:10Do you want to continue press y or n:y
enter your choice:Menu 1.insertion 2. Deletion 3.display:3
20 30 Do you want to continue press y or n:y
enter your choice:Menu 1.insertion 2. Deletion 3.display:2
deleted element is:20Do you want to continue press y or n:y
enter your choice:Menu 1.insertion 2. Deletion 3.display:2
Do you want to continue press y or n:y
enter your choice:Menu 1.insertion 2. Deletion 3.display:3
queue is emptyDo you want to continue press y or n:n

```

9.

### **AIM**

Conversion of Infix expression to Postfix

### **PROGRAM**

```
#include<iostream>
#define SIZE 20
using namespace std;
class stacks
{
    char a[SIZE];
    int top;
public:
    stacks( )
    {
        top=-1;
    }
    void push(char ch);
    char pop( );
    bool isempty( )
    {
        if(top==-1)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
}
```



```

        }
    }
    char top_st( )
    {
        if(top==-1)
        {
            return NULL;
        }
        else
        {
            return a[top];
        }
    }
};

void stacks::push(char ch)
{
    if(top==(SIZE-1))
    {
        cout<<"stack is full";
        return;
    }
    else
    {
        top++;
        a[top]=ch;
    }
}

char stacks::pop( )
{

```

```

        if(top==-1)
        {
            return NULL;
        }
        else
        {
            char data=a[top];
            top--;
            return data;
        }
    }
}

class conv
{
    char in[SIZE];
    char pos[SIZE];
    stacks obj;
    public:
        void read( );
        void display( );
        void convert( );
        int priority(char ch);
};

void conv::read( )
{
    cout<<"\n Enter the infix expression :";
    cin>>in;
}

void conv::display( )

```

```

{
    cout<<"\n Infix expression : "<<in<<"\n";
    cout<<"\n Postfix expression : "<<pos<<"\n";
}
int conv::priority(char ch)
{
    if(ch=='^')
    {
        return 3;
    }
    else if((ch=='%')||(ch=='*')||(ch=='/'))
    {
        return 2;
    }
    else if((ch=='+')||(ch=='-'))
    {
        return 1;
    }
    else
    {
        return 0;
    }
}

void conv::convert( )
{
    char ch,g,t;
    int i=0,j=0;
    while(in[i]!='\0')

```

```

{
    ch=in[i];
    if(in[i]=='(')
    {
        obj.push(in[i]);
    }
    else if((in[i]=='%')||(in[i]=='/')||(in[i]=='*')||(in[i]=='+')||(in[i]=='-'
')||(in[i]=='^'))
    {
        if(!obj.isEmpty( ))
        {
            t=obj.top_st( );
            while((priority(t))>=(priority(ch)))
            {
                pos[j]=obj.pop();
                j++;
                if(!obj.isEmpty( ))
                {
                    t=obj.top_st( );
                }
            }
            else
            {
                break;
            }
        }
        obj.push(in[i]);
    }
    else if(ch=='')

```

```

    {
        if(!obj.isEmpty( ))
        {
            while(obj.top_st( )!='(')
            {
                pos[j]=obj.pop( );
                j++;
            }
            t=obj.pop( );
        }
        else
        {
            break;
        }
    }
    else
    {
        pos[j]=ch;
        j++;
    }

    i++;
}
while(obj.isEmpty( )==false)
{
    pos[j]=obj.pop( );
    j++;
}
pos[j]='\0';
}

```

```
int main( )
{
    conv c;
    c.read( );
    c.convert( );
    c.display( );
    return 0;
}
```

## **OUTPUT**

Enter the infix expression : (a+b)\*c/d+e^f/g

Infix expression : (a+b)\*c/d+e^f/g

Postfix expression : ab+c\*d/ef^g/+

**10.**

**AIM**

Write a program to perform operations on a Circular Queue

**PROGRAM**

```
#include<iostream>

using namespace std;

#define MAXSIZE 5

class queue
{
    int q[10];

    int rear,front;

    public:

    queue( )
    {
        rear=-1;

        front=-1;

    }

    void insert( );

    void delet( );

    void display( );

};
```

```

void queue::insert( )
{
    int item;

    if(front==(rear+1)%MAXSIZE)
    {
        cout<<"queue is over flow:";

    }
    else
    {
        cout<<"enter the element:";

        cin>>item;

        rear=(rear+1)%MAXSIZE;

        q[rear]=item;

        if(front==-1)
        {
            front++;

        }

    }
}

void queue::delet( )

```



```

{
    if(front==-1)
    {
        cout<<"queue is empty:";
    }
    else
    {
        cout<<"deleted element is"<<q[front];
        if(front==rear)
        {
            front=rear=-1;
        }
        else
        {
            front=(front+1)%MAXSIZE;
        }
    }
}

void queue::display( )
{

```

```

int i;

cout<<"\n";

if(front==-1)

{

    cout<<"queue is empty:";

}

else if(rear>=front)

{

    for(i=front;i<=rear;i++)

    {

        cout<<q[i];

        cout<<"\t";

    }

}

else

{

    for(i=front;i<=MAXSIZE-1;i++)

    {

        cout<<q[i]<<"\t";

    }

}

```

```

        for(i=0;i<=rear;i++)
        {
            cout<<q[i]<<"\t";
        }
    }
}

int main( )
{
    queue cq;
    int ch;
    do
    {
        cout<<"\n";
        cout<<"menu: \n 1.insert \n 2.delete \n 3.display \n 4.exit \n";
        cin>>ch;
        switch(ch)
        {
            case 1:
                cq.insert();
                break;

```

```
        case 2:
            cq.delet( );
            break;
        case 3:
            cq.display( );
            break;
        case 4:
            break;
    }
}
while(ch!=4);
return 0;
}
```

## **OUTPUT**

Menu:

1.Insert

2.Delete

3.Display

4.Exit

1

Enter the element: 22

Menu:

1.Insert

2.Delete

3.Display

4.Exit

1

Enter the element: 3

Menu:

1.Insert

2.Delete

3.Display

4.Exit

1

Enter the element: 54

Menu:

1.Insert

2.Delete

3.Display

4.Exit

3

22     3     54

Menu:

1.Insert

2.Delete

3.Display

4.Exit

2

Deleted element is 22

Menu:

1.Insert

2.Delete

3.Display

4.Exit

3

3     54

Menu:

1.Insert

2.Delete

3.Display

4.Exit

4

**11.**

**AIM**

Singly Linked List

**PROGRAM**

```
#include<iostream>

using namespace std;

struct node

{

    int info;

    node *next;

};

node *start;

class slink

{

    public:

    void insertbeg( );

    void insertend( );

    void insertspec( );

    void display( );

    void delbeg( );
```



```

        void delend();

        void delspe( );

};

void slink::insertbeg( )

{

    node *ptr;

    ptr=new node;

        cout<<"enter the limit";

    cin>>ptr->info;

    if(start==NULL)

    {

        ptr->next=NULL;

    }

    else

    {

        ptr->next=start;

    }

    start=ptr;

}

void slink::insertend( )

```

```

{
    node *ptr,*loc;

    ptr=new node;

    cout<<"enter the item";

    cin>>ptr->info;

    ptr->next=NULL;

    if(start==NULL)
    {
        start=ptr;
    }
    else
    {
        loc=start;

        while(loc->next!=NULL)
        {
            loc=loc->next;
        }

        loc->next=ptr;
    }
}

```

```

void slink::insertspec( )
{
    node *ptr,*temp;

    int i,loc;

    ptr=new node;

    cout<<"enter the element";

    cin>>ptr->info;

    cout<<"enter the position";

    cin>>loc; temp=start;

    for(i=0;i<loc-1;i++)

        {

            temp=temp->next;

        }

    ptr->next=temp->next;

    temp->next=ptr;
}

void slink::delbeg( )
{

    node *ptr;

    if(start==NULL)

```

```

        {

            cout<<"List is empty";

            return;

        }

    else

    {

        ptr=start;

        start=start->next;

        delete ptr;

    }

}

void slink::delend()

{

    node *ptr,*loc;

    if(start==NULL)

    {

        cout<<"List is empty";

        return;

    }

    else

```

```

        {
            ptr=start;
            while(ptr->next!=NULL)
            {
                loc=ptr;
                ptr=ptr->next;
            }
            loc->next=NULL;
            delete ptr;
        }
    }

void slink::delspe( )
{
    node *ptr,*save;
    int loc,i;
    cout<<"Enter the position";
    cin>>loc;
    ptr=start;
    for(i=0;i<loc;i++)
    {

```

```

        save=ptr; ptr=ptr->next;

    }

    save->next=ptr->next;

    delete ptr;
}

void slink::display( )
{
    node *ptr;

    ptr=start;

    while(ptr!=NULL)
    {

        cout<<ptr->info;

        ptr=ptr->next;

    }
}

int main( )
{

    slink s;

    int ch,n;

    do

```

```

{

    cout<<"Enter your choice \n 1. Insert at beginning \n 2. Insert
at end \n 3.Insert at specified position\n 4.Delete from beginning\n 5.Delete from
end\n 6.Delete at specified position\n7.Display";

    cin>>ch;

    switch(ch)
    {

        case 1 : s.insertbeg( );

                break;

        case 2: s.insertend( );

                break;

        case 3: s.insertspec( );

                break;

        case 4: s.delbeg( );

                break;

        case 5: s.delend( );

                break;

        case 6: s.delspe( );

                break;

        case 7: s.display( );

                break;
    }
}

```

```
        default:
            cout<<"invalid entry";
        }
        cout<<"do you want to continue if yes press(1)";
        cin>>n;
    } while(n==1);
return 0;
}
```

## **OUTPUT**

Enter your choice :

1. Insert at beginning
2. Insert at end
- 3.Insert at specified position
- 4.Delete from beggining
- 5.Delete from end
- 6.Delete at specified position



7.Display

1

Enter the limit : 3

Do you want to continue ,If yes press(1) : 1

Enter your choice :

1. Insert at beginning

2. Insert at end

3.Insert at specified position

4.Delete from beginning

5.Delete from end

6.Delete at specified position

7.Display

1

Enter the limit : 5

Do you want to continue, If yes press(1) : 1

Enter your choice :

1. Insert at beginning

2. Insert at end

3.Insert at specified position

4.Delete from beginning

5.Delete from end

6.Delete at specified position

7.Display

7

5      3

Do you want to continue, If yes press(1) : 1

Enter your choice :

1. Insert at beginning

2. Insert at end

3.Insert at specified position

4.Delete from beginning

5.Delete from end

6.Delete at specified position

7.Display

2

Enter the item : 6

Do you want to continue ,If yes press(1) : 1

Enter your choice :

1. Insert at beginning

2. Insert at end

3.Insert at specified position

4.Delete from beginning

5.Delete from end

6.Delete at specified position 7.Display

7

5      3      6

Do you want to continue,If yes press(1) : 1

Enter your choice :

1. Insert at beginning

2. Insert at end

3.Insert at specified position

4.Delete from beginning

5.Delete from end

6.Delete at specified position

7.Display

3

enter the element

2

enter the position

1

Do you want to continue, If yes press(1) : 1

Enter your choice :

1. Insert at beginning
2. Insert at end
- 3.Insert at specified position
- 4.Delete from begginning
- 5.Delete from end
- 6.Delete at specified position
- 7.Display

7

5      2      3      6

Do you want to continue, If yes press(1) : 1

Enter your choice :

1. Insert at beginning
2. Insert at end
- 3.Insert at specified position
- 4.Delete from begginning
- 5.Delete from end
- 6.Delete at specified position
- 7.Display

1

Enter the limit : 1

Do you want to continue, If yes press(1) : 1

Enter your choice :

1. Insert at beginning
2. Insert at end
3. Insert at specified position
4. Delete from beginning
5. Delete from end
6. Delete at specified position
7. Display

7

1      5      2      3      6

Do you want to continue ,If yes press(1) : 1

Enter your choice :

1. Insert at beginning
2. Insert at end
3. Insert at specified position
4. Delete from beginning
5. Delete from end

6.Delete at specified position

7.Display

4

Do you want to continue ,If yes press(1) : 1

Enter your choice :

1. Insert at beginning

2. Insert at end

3.Insert at specified position

4.Delete from beginning

5.Delete from end

6.Delete at specified position

7.Display

7

5      2      3      6

Do you want to continue, If yes press(1) : 1

Enter your choice :

1. Insert at beginning

2. Insert at end

3.Insert at specified position

4.Delete from beginning

5.Delete from end

6.Delete at specified position

7.Display

5

Do you want to continue, If yes press(1) : 1

Enter your choice :

1. Insert at beginning

2. Insert at end

3.Insert at specified position

4.Delete from beginning

5.Delete from end

6.Delete at specified position

7.Display

7

5      2      3

Do you want to continue, If yes press(1) : 1

Enter your choice :

1. Insert at beginning

2. Insert at end

3.Insert at specified position

4.Delete from beginning

5.Delete from end

6.Delete at specified position

7.Display

6

Enter the position : 1

Do you want to continue, If yes press(1) : 1

Enter your choice :

1. Insert at beginning

2. Insert at end

3.Insert at specified position

4.Delete from beginning

5.Delete from end

6.Delete at specified position

7.Display

7

5      3

Do you want to continue, If yes press(1) : 2



## 12.

### **AIM**

Circular Linked list

### **PROGRAM**

```
#include <iostream>
using namespace std;
struct node
{
    int info;
    node *next;
};
node *start, *last;
class clink
{
public:
    void insertbeg();
    void insertend();
    void insertspec();
    void deletbeg();
    void deletend();
    void deletspec();
    void display();
};
void clink::insertbeg()
{
    node *ptr;
    ptr = new node;
    cout << "enter the item\n";
```

```

    cin >> ptr->info;
    if (start == NULL)
    {
        start = ptr;
        ptr->next = start;
        last = ptr;
    }
    else
    {
        ptr->next = start;
        start = ptr;
        last->next = ptr;
    }
}
void clink::insertend()
{
    node *ptr, *loc;
    ptr = new node;
    cout << "\nEnter the item\n";
    cin >> ptr->info;
    ptr->next = start;
    if (start == NULL)
    {
        start = ptr;
        last->next = ptr;
    }
    else
    {
        last->next = ptr;
    }
}

```

```

        last = ptr;
    }
}
void clink::insertspec()
{
    node *ptr, *temp;
    int loc, i;
    ptr = new node;
    cout << "\n Enter the element\n";
    cin >> ptr->info;
    cout << "\n Enter the location\n";
    cin >> loc;
    temp = start;
    for (i = 1; i < loc - 1; i++)
    {
        temp = temp->next;
    }
    ptr->next = temp->next;
    temp->next = ptr;
}
void clink::deletbeg()
{
    node *ptr;
    ptr = start;
    if (start == NULL)
    {
        cout << "\n List is empty\n";
        return;
    }
}

```

```

else if (start == last)
{
    start = last = NULL;
    delete ptr;
}
else
{
    start = start->next;
    last->next = start;
    delete ptr;
}
}
void clink::deletend()
{
    node *ptr, *temp;
    ptr = start;
    if (start == NULL)
    {
        cout << "\n List is empty\n";
        return;
    }
    else if (ptr->next == start)
    {
        start = last = NULL;

        delete ptr;
    }
    else
    {

```

```

        while (ptr->next != start)
        {
            temp = ptr;
            ptr = ptr->next;
        }
        temp->next = start;
        last = temp;
        delete ptr;
    }
}

void clink::deletspec()
{
    node *ptr, *save;
    int loc, i;
    cout << "\n Enter the position\n";
    cin >> loc;
    ptr = start;
    for (i = 1; i < loc; i++)
    {
        save = ptr;
        ptr = ptr->next;
    }
    save->next = ptr->next;
    delete ptr;
}

void clink::display()
{
    node *ptr;
    if (start == NULL)

```

```

    {
        cout << "\n list is empty\n";
        return;
    }
else

    {
        ptr = start;
        while (ptr->next != start)
        {
            cout << ptr->info << "\t";
            ptr = ptr->next;
        }
        cout << ptr->info << "\t";
    }
}
int main()
{
    clink c;
    int ch, n;
    do
    {
        cout << "enter your choice\n 1.insertion at the begening\n 2.insertion at specific
position\n3.insertion at the end\n4.delelete from the beginning\n5.deletete from
end\n6.deletete from specific position\n7.display";
        cin >> ch;
        switch (ch)
        {
            case 1:

```

```

        c.insertbeg();
        break;
case 2:
        c.insertspec();
        break;
case 3:
        c.insertend();
        break;
case 4:
        c.deletbeg();
        break;
case 5:
        c.deletend();
        break;
case 6:
        c.deletspec();

        break;
case 7:
        c.display();
        break;
case 8:
        break;
    }
    cout << "\ndoyou want to continue press(1)\n";
    cin >> n;
} while (n == 1);
return 0;
}

```

## **OUTPUT**

enter your choice

- 1.insertion at the begening
- 2.insertion at specific position
- 3.insertion at the end
- 4.delelete from the beginning
- 5.deletete from end
- 6.deletete from specific position
- 7.display1

enter the item

10

doyou want to continue press(1)

1

enter your choice

- 1.insertion at the begening
- 2.insertion at specific position
- 3.insertion at the end
- 4.delelete from the beginning
- 5.deletete from end
- 6.deletete from specific position
- 7.display1

enter the item

20

doyou want to continue press(1)

1

enter your choice

- 1.insertion at the begening
- 2.insertion at specific position
- 3.insertion at the end
- 4.delelete from the beginning
- 5.deletete from end
- 6.deletete from specific position
- 7.display3



enter the item

30

doyou want to continue press(1)

1

enter your choice

1.insertion at the begening

2.insertion at specific position

3.insertion at the end

4.delelete from the beginning

5.deletete from end

6.deletete from specific position

7.display7

20 10 30

doyou want to continue press(1)

1

enter your choice

1.insertion at the begening

2.insertion at specific position

3.insertion at the end

4.delelete from the beginning

5.deletete from end

6.deletete from specific position

7.display2

enter the element

40

enter the location

2

doyou want to continue press(1)

1

enter your choice

1.insertion at the begening

2.insertion at specific position  
3.insertion at the end  
4.delete from the beginning  
5.delete from end  
6.delete from specific position  
7.display7  
20 40 10 30  
doyou want to continue press(1)  
1

enter your choice  
1.insertion at the begening  
2.insertion at specific position  
3.insertion at the end  
4.delete from the beginning  
5.delete from end  
6.delete from specific position  
7.display4

doyou want to continue press(1)  
1  
enter your choice  
1.insertion at the begening  
2.insertion at specific position  
3.insertion at the end  
4.delete from the beginning  
5.delete from end  
6.delete from specific position  
7.display7  
40 10 30  
doyou want to continue press(1)  
2

**13.**

**AIM**

Doubly linked list

**PROGRAM**

```
#include<iostream>
using namespace std;
struct node
{
    int info;
    node *next;
    node *prev;
};
node *start,*last;

class doublylink
{
public:
    void insert_beg( );
    void insert_end( );
    void insert_spec( );
    void del_beg( );
    void del_end( );
    void del_spec( );
    void display( );
};
void doublylink::insert_beg( )
{
```

```

node*ptr;
ptr=new node;
cout<<"enter the element";
cin>>ptr->info;
ptr->prev=NULL;
if(start==NULL)
{
    ptr->next=NULL;
    start=ptr;
    last=ptr;
}
else
{
    start->prev=ptr;
    ptr->next=start;
    start=ptr;
}
}
void doublylink::insert_end( )
{
    node*ptr;
    ptr=new node;
    cout<<"enter the element";
    cin>>ptr->info;
    ptr->next=NULL;
    if(start==NULL)
    {
        ptr->prev=NULL;
        start=ptr;
    }
}

```

```

        last=ptr;
    }
    else
    {
        last->next=ptr;
        ptr->prev=last;
        last=ptr;
    }
}

void doublylink::insert_spec( )
{
    node *ptr,*save,*temp;
    int i,loc;
    ptr=new node;
    temp=start;
    cout<<"enter the element";
    cin>>ptr->info;
    cout<<"enter the location";
    cin>>loc;
    for(i=1;i<loc;i++)
    {
        save=temp;
        temp=temp->next;
    }
    save->next=ptr;
    ptr->prev=save;
    ptr->next=temp;
    temp->prev=ptr;
}

```

```

void doublylink::del_beg( )
{
    node*ptr;
    if(start==NULL)
    {
        cout<<"list is empty";
        return;
    }
    else
    {
        ptr=start;
        start=start->next;
        start->prev=NULL;
        delete ptr;
    }
}

void doublylink::del_end( )
{
    node*ptr,*loc;
    if(start==NULL)
    {
        cout<<"list is empty";
        return;
    }
    else
    {
        ptr=last;
        last=last->prev;
        last->next=NULL;
    }
}

```

```

        delete ptr;
    }
}

void doublylink::del_spec( )
{
    node *ptr,*save,*temp;
    int i,loc;
    ptr=start;
    cout<<"enter the location";
    cin>>loc;
    if(start==NULL)
    {
        cout<<"list is empty";
        return;
    }
    else
    {
        for(i=1;i<loc;i++)
        {
            temp=ptr;
            ptr=ptr->next;
        }
        save=ptr->next;
        temp->next=save;
        save->prev=temp;
        delete ptr;
    }
}

void doublylink::display( )

```

```

{
    node *ptr;
    ptr=start;
    while(ptr!=NULL)
    {
        cout<<ptr->info<<"->";
        ptr=ptr->next;
    }
}

int main( )
{
    doublylink dl;
    int ch,n;
    do
    {
        cout<<"enter your choice\n 1.insert at beginning\n 2.insert at end\n
        3.insert at specific position\n 4.delete from beginning\n 5.delete from
        end\n 6.delete from specific position\n 7.display";
        cin>>ch;
        switch(ch)
        {
            case 1:
                dl.insert_beg( );
                break;
            case 2:
                dl.insert_end( );
                break;
            case 3:
                dl.insert_spec( );

```



```

        break;
    case 4:
        dl.del_beg( );
        break;
    case 5:
        dl.del_end( );
        break;
    case 6:
        dl.del_spec( );
        break;
    case 7:
        dl.display( );
        break;
    default:
        cout<<"invalid entry";

    }

    cout<<"do you want to continue if yes press 1";
    cin>>n;
}
while(n==1);
return 0;
}

```

## **OUTPUT**

enter your choice

1.insert at beginning

2.insert at end

3.insert at specific position

4.delete from beginning

5.delete from end

6.delete from specific position

7.display 1

enter the element 2

do you want to continue if yes press 1 : 1

enter your choice

1.insert at beginning

2.insert at end

3.insert at specific position

4.delete from beginning

5.delete from end

6.delete from specific position

7.display 2

enter the element 6

do you want to continue if yes press 1: 1

enter your choice

1.insert at beginning

2.insert at end

3.insert at specific position

4.delete from beginning

5.delete from end

6.delete from specific position

7.display 3

enter the element 8

enter the location 2

do you want to continue if yes press 1: 1

enter your choice

1.insert at beginning

2.insert at end

3.insert at specific position

4.delete from beginning

5.delete from end

6.delete from specific position

7.display 7

2->8->6->do you want to continue if yes press 1: 1

enter your choice

1.insert at beginning

2.insert at end

3.insert at specific position

4.delete from beginning

5.delete from end

6.delete from specific position

7.display 4

do you want to continue if yes press 1: 1

enter your choice

1.insert at beginning

2.insert at end

3.insert at specific position

4.delete from beginning

5.delete from end

6.delete from specific position

7.display 7

8->6->do you want to continue if yes press 1: 1

enter your choice

1.insert at beginning

2.insert at end

3.insert at specific position

4.delete from beginning

5.delete from end

6.delete from specific position

7.display 5

do you want to continue if yes press 1 : 1

enter your choice

1.insert at beginning

2.insert at end

3.insert at specific position

4.delete from beginning

5.delete from end

6.delete from specific position

7.display 7

8->do you want to continue if yes press 1 : 2

**14.**

**AIM**

Implement tree traversal

**PROGRAM**

```
#include<iostream>
using namespace std;
struct node
{
    int data;
    struct node *left;
    struct node *right;
};
void preorder(struct node *root)
{
    if(root==NULL)
    {
        return;
    }
    cout<<root->data<<"\t";
    preorder(root->left);
    preorder(root->right);
}
void inorder(node *root)
{
    if(root==NULL)
    {
```

```

        return;
    }
    inorder(root->left);
    cout<<root->data<<"\t";
    inorder(root->right);
}

void postorder(node *root)
{
    if(root==NULL)
    {
        return;
    }
    postorder(root->left);
    postorder(root->right);
    cout<<root->data<<"\t";
}

node *insert(node *root,int data)
{
    if(root==NULL)
    {
        root=new node( );
        root->data=data;
        root->left=root->right=NULL;
    }
    else if(data<=root->data)
    {
        root->left=insert(root->left,data);
    }
    else

```

```

        {
            root->right=insert(root->right,data);
        }
        return root;
    }
int main( )
{
    node *root=NULL;
    root=insert(root,2);
    root=insert(root,10);
    root=insert(root,22);
    root=insert(root,8);
    root=insert(root,11);
    root=insert(root,30);
    cout<<"Preorder";
    preorder(root);
    cout<<"\n";
    cout<<"Inorder";
    inorder(root);
    cout<<"\n";
    cout<<"Postorder";
    postorder(root);
    cout<<"\n";
    return 0;
}

```

## **OUT PUT**

Preorder 2 10 8 22 11 30

Inorder 2 8 10 11 22 30

Postorder 8 11 30 22 10 2



**15.**

**AIM**

Merge to sort linked list

**PROGRAM**

```
#include <iostream>
using namespace std;
struct Node
{
    int data;
    struct Node *next;
};
struct Node *newNode(int key)
{
    struct Node *temp = new Node;
    temp->data = key;
    temp->next = NULL;
    return temp;
}
void printList(struct Node *node)
{
    while (node != NULL)
    {
        cout << node->data << "->";

        node = node->next;
    }
}
struct Node *mergeUtil(struct Node *h1, struct Node *h2)
{

```

```

if (!h1->next)
{
    h1->next = h2;
    return h1;
}
struct Node *curr1 = h1, *next1 = h1->next;
struct Node *curr2 = h2, *next2 = h2->next;
while (next1 && next2)
{
    if ((curr2->data) > (curr1->data) && (curr2->data) < (next1->data))
    {
        next2 = curr2->next;
        curr1->next = curr2;
        curr2->next = next1;
        curr1 = curr2;
        curr2 = next2;
    }
    else

    {
        if (next1->next)
        {
            next1 = next1->next;
            curr1 = curr1->next;
        }
        else
        {
            next1->next = curr2;
            return h1;
        }
    }
}

```

```

    }
    return h1;
}
struct Node *merge(struct Node *h1, struct Node *h2)
{
    if (!h1)
    {
        return h2;
    }
    if (!h2)
    {
        return h1;
    }
    if (h1->data < h2->data)
    {
        return mergeUtil(h1, h2);
    }
    else
    {
        return mergeUtil(h2, h1);
    }
}
int main()
{
    struct Node *head1 = newNode(1);
    head1->next = newNode(3);
    head1->next->next = newNode(5);
    struct Node *head2 = newNode(0);
    head2->next = newNode(2);
    head2->next->next = newNode(4);

```

```
    struct Node *mergedhead = merge(head1, head2);  
    printList(mergedhead);  
    return 0;  
}
```

## **OUTPUT**

0->1->2->3->4->5->

# **DATABASE MANAGEMENT SYSTEM**

## SQL-1

***Q. Create table student with fields sno, sname, sex, mark with sno as primary key and assign suitable constraints for each attribute.***

Create table student(sno int primary key, sname varchar(20) NOT NULL, sex char, mark int);

Insert five records into the table

```
insert into student values(1,'Ammu','F',50);  
INSERT 0 1
```

```
insert into student values(2,'Justin','M',45);  
INSERT 0 1
```

```
insert into student values(3,'John','M',40);  
INSERT 0 1
```

```
insert into student values(4,'Dilna','F',50);  
INSERT 0 1
```

```
insert into student values(5,'Diyana','F',50);  
INSERT 0 1
```

```
select *from student;
```

sno	sname	sex	mark
1	Ammu	F	50
2	Justin	M	45
3	John	M	40
4	Dilna	F	50
5	Diyana	F	50

(5 rows)

***A. Alter the table by adding one more field rank.***

```
alter table student add rank int;  
ALTER TABLE
```

***B. Display all boy students with their name.***

```
select sname from student where sex='M';
```

```
sname
```

```
-----
```

```
Justin
```

```
John
```

(2 rows)

***C. Find the average mark.***

```
select avg(mark) from student;
```

```
avg
```

```
-----
```

```
47.0000000000000000
```

(1 row)

***D. Create a query to display the sno and sname for all students who got more than the average mark.***

```
select sno,sname from student where mark>(select avg(mark) from student);
```

```
sno | sname
```

```
-----+-----
```

```
1 | Ammu
```

```
4 | Dilna
```

```
5 | Diyana
```

(3 rows)

***E. Sorts the results in descending order of the mark.***

select sno,sname from student where mark>(select avg(mark) from student)order  
by mark desc;

sno	sname
1	Ammu
4	Dilna
5	Diyana

(3 rows)

***F. Display all girl students names for those who have marks greater than 20  
and less than 40.***

select \*from student where(sex='F')and(mark not between 40 and 20);

sno	sname	sex	mark	rank
1	Ammu	F	50	
4	Dilna	F	50	
5	Diyana	F	50	

(3 rows)



## SQL 2

**Q. Create table department with fields ename,salary,dno,dname and place with dno as primary key.**

```
create table dept(ename varchar(20),salary int,dno int primary key,dname
    varchar(20),place varchar(20));
CREATE TABLE
```

**A: insert 5 records**

```
insert into dept values('abi',50000,101,'purchase','kannur');
INSERT 0 1
```

```
insert into dept values('rithul',80000,102,'marketing','calicut');
INSERT 0 1
```

```
insert into dept values('vaishnav',10000,103,'sale','kochi');
INSERT 0 1
```

```
insert into dept values('ashika',1200,104,'accounts','trivandram');
INSERT 0 1
```

```
insert into dept values('anu',12000,105,'accounts','kollam');
INSERT 0 1
```

```
select *from dept;
```

ename	salary	dno	dname	place
abi	50000	101	purchase	kannur
rithul	80000	102	marketing	calicut
vaishnav	10000	103	sale	kochi
ashika	1200	104	accounts	trivandram
anu	12000	105	accounts	kollam

(5 rows)

**B:Rename the field place with city .**

alter table dept rename place to city;  
ALTER TABLE

select \*from dept;

ename	salary	dno	dname	city
abi	50000	101	purchase	kannur
rithul	80000	102	marketting	calicut
vaishnav	10000	103	sale	kochi
ashika	1200	104	accounts	trivandram
anu	12000	105	accounts	kollam

(5 rows)

**C: Display the employees who get salary more than 6000 and less than 10000**

select ename from dept where salary between 6000 and 10000;  
ename  
-----  
vaishnav  
(1 row)

**D:Display total salary of the organisation.**

select sum(salary)from dept;  
sum  
-----  
153200  
(1 row)

**E: Display ename for those who are getting salary in between 5000 and 10000**

```
SELECT ename FROM dept WHERE salary BETWEEN 5000 and 10000;  
ename
```

-----

```
vaishnav  
(1 row)
```

**F: Create view name 'star' with field ename,salary and place.**

```
create view star as (select ename,salary,city from dept);  
CREATE VIEW
```

**G: Display ename and salary with salary rounded with 10 digits '\*'. \***

```
select ename,round(salary,10)from dept;
```

ename	round
abi	50000.0000000000
rithul	80000.0000000000
vaishnav	10000.0000000000
ashika	1200.0000000000
anu	12000.0000000000

(5 rows)

### SQL -3

***Create a table department with fields dno,dname,dmanager,and place with dno as primary key.***

***Create a table emp with fields eno,ename,job,dno,salary,with eno as primary key.Set dno as foreign key.***

```
create table department (dno int primary key,dname varchar(30),place  
varchar(30),dmanager varchar(20),);  
CREATE TABLE
```

***Create a table emp with fields eno,ename,job,dno,salary,with eno as primary key.Set dno as foreign key.***

```
create table emp (eno int primary key,ename varchar(30),job varchar(30),salary  
int,dno int ,foreign key(dno) references department(dno));  
CREATE TABLE
```

***Insert five records into each table.***

```
insert into department values(1,'Sales','Kannur','Manu');  
INSERT 0 1
```

```
insert into department values(2,'Marketing','Kasargode','Arjun');  
INSERT 0 1
```

```
insert into department values(3,'Marketing','Kollam','Suresh');  
INSERT 0 1
```

```
insert into department values(4,'Sales','Calicut','Rolex');  
INSERT 0 1
```

```
insert into department values(5,'Sales','Calicut','Dilli');  
INSERT 0 1
```

```

insert into emp values(1,'Dulqer','Sales',10000,4);
INSERT 0 1
Department=#
insert into emp values(2,'Ramcharan','Marketing',11000,1);
INSERT 0 1
Department=#
insert into emp values(3,'Suresh Gopi','Sales',11000,2);
INSERT 0 1
Department=#
insert into emp values(4,'Dharmajan','Sales',118800,1);
INSERT 0 1
Department=#
insert into emp values(5,'Vineeth','Sales',18800,5);
INSERT 0 1

```

### ***1.Display the ename and salary,salary with ascending order***

```
select ename ,salary from emp order by salary asc;
```

ename	salary
Dulqer	10000
Ramcharan	11000
Suresh Gopi	11000
Vineeth	18800
Dharmajan	118800

(5 rows)

### ***2.Display ename and salary for eno = 20***

```
select ename ,salary from emp where eno = 20;
```

ename	salary
-------	--------

(0 rows)

### ***3.Display the manager for the accounting Department***

```
select dmanager from department where dname = 'accounting';  
dmanager
```

```
-----  
(0 rows)
```

### ***4.Display the name,salary and manager of all employees who are getting salary > 5000***

```
select emp.ename,emp.salary,department.dmanager from emp inner join  
department on emp. dno = department.dno where emp.salary >5000;
```

ename	salary	dmanager
Dulqer	10000	Rolex
Ramcharan	11000	Manu
Suresh Gopi	11000	Arjun
Dharmajan	118800	Manu
Vineeth	18800	Dilli

```
(5 rows)
```

### ***5.Write the queries using various group functions.***

```
select count (*) from emp;  
count
```

```
-----  
5  
(1 row)
```

```
select distinct job from emp;  
job
```

```
-----  
Marketing  
Sales  
(2 rows)
```

```
select count (distinct job) from emp;
count
-----
      2
(1 row)
```

```
select max (salary) from emp;
max
-----
118800
(1 row)
```

```
select avg (salary) from emp;
avg
-----
33920.000000000000000000
(1 row)
```

***6. Write the queries using various Number functions.***

```
select abs(salary) from emp where ename = 'Vineeth';
abs
-----
18800
(1 row)
```

```
select greatest(salary) from emp ;
greatest
-----
10000
11000
11000
118800
18800
(5 rows)
```

## SQL-4

**Q.Create a table employee with fields eno,ename,job,manager and salary with eno as primary key.Create a sequence to be used with employee table's**

primary key column.The sequence should start at 60 and maximum value of 200,increment by 10 numbers.

```
create sequence emp_seq minvalue 0 maxvalue 200 increment 10 start 60;
```

```
CREATE SEQUENCE
```

```
create table emp(eno int default nextval('emp_seq')primary key,ename  
varchar(30),job varchar(20),manager varchar(30),salary int);
```

```
CREATE TABLE
```

A) Insert five records

```
insert into emp values(60,'ashika','hro','anusree',60000);
```

```
INSERT 0 1
```

```
insert into emp values(70,'ashna','ass.manager','ammu',90000);
```

```
INSERT 0 1
```

```
insert into emp values(80,'surya','supervisor','aju',60000);
```

```
INSERT 0 1
```

```
insert into emp values(90,'prannya','clerk','yadhu',42000);
```

```
INSERT 0 1
```

```
insert into emp values(100,'thara','accountant','yachu',56000);
```

```
INSERT 0 1
```

```
select *from emp;
```

eno	ename	job	manager	salary
-----	-----	-----	-----	-----
60	ashika	hro	anusree	60000
70	ashna.	ass.manager	ammu	90000
80	surya	supervisor.	aju	60000
90	prannya	clerk	yadhu	42000
100	thara	accountant	yachu	56000

(5 rows)



**A) Display ename,salary from employee where salary more than average salary**

```
select ename,salary from emp where salary>(select avg(salary)from emp);
```

```
ename | salary
```

```
-----+-----
```

```
ashna | 90000
```

```
(1 row)
```

**B) Add 20%DA extra salary to all employees,label the column as 'new\_salary'**

```
alter table emp add new_salary int;
```

```
ALTER TABLE
```

```
select *from emp;
```

```
eno | ename | job | manager | salary | new_salary
```

```
----+-----+-----+-----+-----+-----
```

```
60 | ashika | hro | anusree | 60000 |
```

```
70 | ashna | ass.manager | ammu | 90000 |
```

```
80 | surya | supervisor | aju | 60000 |
```

```
90 | prannya | clerk | yadhu | 42000 |
```

```
100 | thara | accountant | yachu | 56000 |
```

```
(5 rows)
```

```
update emp set new_salary=salary+(salary*0.2);
```

```
UPDATE 5
```

```
select *from emp;
```

```
eno | ename | job | manager | salary | new_salary
```

```
----+-----+-----+-----+-----+-----
```

```
60 | ashika | hro | anusree | 60000 | 72000
```

```
70 | ashna | ass.manager | ammu | 90000 | 108000
```

```
80 | surya | supervisor | aju | 60000 | 72000
```

```
90 | prannya | clerk | yadhu | 42000 | 50400
```

```
100 | thara | accountant | yachu | 56000 | 67200
```

```
(5 rows)
```

**C) Create a query to display the eno and ename for all employees who earn more than average salary,sort the result in descending order of salary**

```
select eno,ename from emp where salary>(select avg(salary)from emp)order  
by salary desc;
```

```
eno | ename
```

```
-----+-----
```

```
70 | ashna
```

```
(1 row)
```

**D) Create a view called emp\_view based on the eno and ename from employee.Change the heading for the ename to employee.**

```
create view emp_view(eno,ename)as(select eno,ename from emp);
```

```
CREATE VIEW
```

```
select *from emp_view;
```

```
eno | ename
```

```
-----+-----
```

```
60 | ashika
```

```
70 | ashna
```

```
80 | surya
```

```
90 | prannya
```

```
100 | thara
```

```
(5 rows)
```

**E) Write a query that will display the eno and ename for all employee who work,whose name contains a 't'**

```
select eno,ename from emp where ename like '%t%';
```

```
eno | ename
```

```
-----+-----
```

```
100 | thara
```

```
(1 row)
```

## SQL-5

***Q. Create a table department with fields dno,ename,salary ,designation,dname and place with dno as primary key.***

```
create table department(dno int primary key,ename varchar(20),salary float,designation char(20),dname char(20),place char(20));
```

CREATE TABLE

Insert values into the table.

```
insert into department values(1,'Soumya',8000,'Clerk','Office','Kannur');
```

INSERT 0 1

```
insert into department values(2,'Jithesh',7000,'Peon','Office','Kanjangad');
```

INSERT 0 1

```
insert into department values(3,'Rishikesh',15000,'Manager','Management','Thalassery');
```

INSERT 0 1

```
insert into department values(4,'Mukundan',10000,'Pro','Administration','Wayanad');
```

INSERT 0 1

```
insert into department values(5,'Shylaja',5000,'Helper','Refreshment','Iritty');
```

INSERT 0 1

select \*from department;

eno	ename	salary	designation	dname	place
1	Soumya	8000	Clerk	Office	
	Kannur				
2	Jithesh	7000	Peon	Office	
	Kanjangad				
3	Rishikesh	15000	Manager	Management	
	Thalassery				
4	Mukundan	10000	Pro	Administration	
	Wayanad				
5	Shylaja	5000	Helper	Refreshment	Iritty

(5 rows)

***A. write the queries using various character functions in ename field.***

select upper(ename) from department;

```
upper
-----
SOUMYA
JITHESH
RISHIKESH
MUKUNDAN
SHYLAJA
```

(5 rows)

```
select lower(ename) from department;
```

```
lower
-----
soumya
jithesh
rishikesh
mukundan
shylaja
```

(5 rows)

```
select trim(ename) from department;
```

```
trim
-----
Soumya
Jithesh
Rishikesh
Mukundan
Shylaja
```

(5 rows)

***B. Create a query to display the employee number and name for all employees who earn more than the average salary. Sort the results in descending order of salary.***

```
select eno,ename from department where salary>(select avg(salary)from
department)order by salary desc;
```

```
eno |  ename
-----+-----
3  | Rishikesh
4  | Mukundan
```

(2 rows)

***C. Display all employees who got salary between 5000 & 10000.***

select ename from department where salary between 5000 and 10000;

ename

-----

Soumya

Jithesh

Mukundan

Shylaja

(4 rows)

***D. Display ename,salary,designation for those who got salary more than 5000 or his designation is 'Clerk'.***

select ename,salary,designation from department where salary>5000 or designation='Clerk';

ename	salary	designation
-------	--------	-------------

-----+-----+-----

Soumya	8000	Clerk
--------	------	-------

Jithesh	7000	Peon
---------	------	------

Rishikesh	15000	Manager
-----------	-------	---------

Mukundan	10000	Pro
----------	-------	-----

(4 rows)

***E. Display ename and designation those who are not a clerk or manager.***

select ename,designation from department where designation not  
in('Manager','Clerk');

ename	designation
Jithesh	Peon
Mukundan	Pro
Shylaja	Helper

(3 rows)

***F. Display the names of all employees where the third letter of their name is an 'a'.***

select ename from department where ename like '\_\_a%';

ename

-----

(0 rows)

## SQL-6

**Q.Create a table customer with fields cid,cname,d.o.b,place**

```
create table cust(cid int primary key,cname varchar(20),dob date,place  
varchar(20));  
CREATE TABLE
```

**Q.Create a table loan with fields loanno,cid,bname assigning suitable constraints.**

```
create table loan (lno int primary key,cid int references cust(cid),bname  
varchar(20));  
CREATE TABLE
```

**Q.Create a table depositor with accno,cid,bname,balance assigning suitable constraints.**

```
create table dep (accno int primary key,cid int references cust(cid),balance  
float,bname varchar(20));  
CREATE TABLE
```

**Q.Insert 5 records into each table**

```
insert into cust values(1,'sahad','1999-09-22','kannur');  
INSERT 0 1  
insert into cust values(2,'sarath','1999-05-27','kannur');  
INSERT 0 1  
insert into cust values(3,'ammmu','1999-05-07','kochi');  
INSERT 0 1  
insert into cust values(4,'arun','1999-08-22','kochi');  
INSERT 0 1  
insert into cust values(5,'athira','1998-09-11','calicut');  
INSERT 0 1
```



```
select*from cust;
```

cid	cname	dob	place
1	sahad	1999-09-22	kannur
2	sarath	1999-05-27	kannur
3	ammmu	1999-05-07	kochi
4	arun	1999-08-22	kochi
5	athira	1998-09-11	calicut

(5 rows)

```
insert into loan values(01,2,'kannur');
INSERT 0 1
insert into loan values(02,1,'kannur');
INSERT 0 1
insert into loan values(03,3,'kochi');
INSERT 0 1
insert into loan values(04,4,'kochi');
INSERT 0 1
insert into loan values(05,5,'calicut');
INSERT 0 1
```

```
select*from loan;
lno | cid | bname
-----+-----+-----
1 | 2 | kannur
2 | 1 | kannur
3 | 3 | kochi
4 | 4 | kochi
5 | 5 | calicut
(5 rows)
```

```

insert into dep values(10005,1,5000,'kannur');
INSERT 0 1
insert into dep values(10006,2,52000,'kannur');
INSERT 0 1
insert into dep values(10007,2,22000,'kochi');
INSERT 0 1
insert into dep values(10009,3,22000,'kochi');
INSERT 0 1
insert into dep values(10010,4,25000,'kochi');
INSERT 0 1
insert into dep values(10011,5,30000,'calicut');
INSERT 0 1

```

```

select*from dep;
 accno | cid | balance | bname
-----+-----+-----+-----
 10005 |  1 |   5000 | kannur
 10006 |  2 |  52000 | kannur
 10007 |  2 |  22000 | kochi
 10009 |  3 |  22000 | kochi
 10010 |  4 |  25000 | kochi
 10011 |  5 |  30000 | calicut
(6 rows)

```

A: Add one more field amount to loan table .Update each record.Display cname for cid=2.

```

alter table loan add amount float;
ALTER TABLE

```

```

select*from loan;
 lno | cid | bname | amount
-----+-----+-----+-----
  1 |  2 | kannur |
  2 |  1 | kannur |
  3 |  3 | kochi  |
  4 |  4 | kochi  |
  5 |  5 | calicut |    5 rows)

```

```
update loan set amount=5000.00 where cid=1;  
UPDATE 1
```

```
update loan set amount=25000.00 where cid=2;  
UPDATE 1
```

```
update loan set amount=15000.00 where cid=3;  
UPDATE 1
```

```
update loan set amount=7000.00 where cid=4;  
UPDATE 1
```

```
update loan set amount=9000.00 where cid=5;  
UPDATE 1
```

```
select*from loan;  
lno | cid | bname | amount  
-----+-----+-----+-----  
2 | 1 | kannur | 5000  
1 | 2 | kannur | 25000  
3 | 3 | kochi | 15000  
4 | 4 | kochi | 7000  
5 | 5 | calicut | 9000  
(5 rows)
```

```
select cname from cust where cid=2;  
cname  
-----  
sarath  
(1 row)
```

B. Calculate rs.150 extra for all customers having loan. The added loan amount will be display in a new coloumn

```
alter table loan add newamount float;  
ALTER TABLE
```

```
select*from loan;
```

lno	cid	bname	amount	newamount
2	1	kannur	5000	
1	2	kannur	25000	
3	3	kochi	15000	
4	4	kochi	7000	
5	5	calicut	9000	

(5 rows)

```
update loan set newamount=amount+150;  
UPDATE 5
```

```
select*from loan;
```

lno	cid	bname	amount	newamount
2	1	kannur	5000	5150
1	2	kannur	25000	25150
3	3	kochi	15000	15150
4	4	kochi	7000	7150
5	5	calicut	9000	9150

(5 rows)

C:Display loanno ,cname and place of a customer who is residing in kannur city.

```
select loan.lno,cust.cname,cust.place from loan inner join cust on  
loan.cid=cust.cid where cust.place='kannur';
```

lno	cname	place
2	sahad	kannur
1	sarath	kannur

(2 rows)

D.Display all informations from loan table for loanno=2,3,5

```
select*from loan where lno=2 or lno=3 or lno=5;
```

```
lno | cid | bname | amount | newamount
```

```
-----+-----+-----+-----+-----
```

```
2 | 1 | kannur | 5000 | 5150
```

```
3 | 3 | kochi | 15000 | 15150
```

```
5 | 5 | calicut | 9000 | 9150
```

(3 rows)

E:Display all customer who have both loan and deposit.

```
select cust.cname from cust inner join loan on cust.cid=loan.cid inner join dep  
on cust.cid=dep.cid;
```

```
cname
```

```
-----
```

```
sahad
```

```
sarath
```

```
sarath
```

```
ammmu
```

```
arun
```

```
athira
```

(6 rows)