WADIHUDA

WIRAS

WADIHUDA INSTITUTE OF RESEARCH & ADVANCED STUDIES

(Affiliated to Kannur University)

(Affiliated by Kannur University)

Vilayancode (PO), Kannur Dist., Kerala   670504

www.wiraskannur.com

DATA STRUCTURES & DBMS

PRACTICAL RECORD

BCA

4th SEMESTER

2021

WADIHUDA INSTITUTE FOR RESEARCH AND ADVANCED STUDIES (WIRAS)

(Affiliated by Kannur University)

Vilayancode (Po),Kannur (dt),Kerala

Pin Code-670 501

www.wiraskannur.com

Dept. Of Computer Science

DATA STRUCTURES & DBMS

PRACTICAL RECORD

NAME:………

UNIVERSITY REG NO:……………………………………………………………………………………

Certified That This Is a Bonafede Record Practical

Work Done By………………………………………………………

In Wadihuda Institute of Research and Advanced Studies (WIRAS) for the year 2020 to 2021

Faculty in charge	***Head of the Department***	***External Examiners***

Date:

DATA STRUCTURE

DBMS

1.Add two polynomials.

```
#include<iostream.h>
#include<conio.h>
#define MAX 10
int main()
{
int poly1[MAX]={0},poly2[MAX]={0},poly3[MAX]={0};
int i,deg1,deg2,deg3;
clrscr();
cout<<"\nEnter the degree of polynomial 1:";
cin>>deg1;
cout<<"\nEnter the degree of polynomial 2:";
cin>>deg2;
cout<<"\nFor polynomial 1";
for(i=0;i<=deg1;i++)
{
cout<<"\nEnter the coefficient of exponent"<<i<<":";
cin>>poly1[i];
}
cout<<"\nFor polynomial 2";
for(i=0;i<=deg2;i++)
{
cout<<"\nEnter the coefficient for exponent"<<i<<":";
cin>>poly2[i];
}
deg3=(deg1>deg2)?deg1:deg2;
for(i=0;i<=deg3;i++)
{
poly3[i]=poly1[i]+poly2[i];
}
cout<<"\nPolynomial 1:";
for(i=deg1;i>0;i--)
{
cout<<poly1[i]<<"x^"<<i<<"+";
}
cout<<poly1[0];
cout<<"\nPolynomial 2:";
for(i=deg2;i>0;i--)
{
cout<<poly2[i]<<"x^"<<i<<"+";
}
cout<<poly2[0];
cout<<"\n RESULT \n";
for(i=deg3;i>0;i--)
{
cout<<poly3[i]<<"x^"<<i<<"+";
}
cout<<poly3[0];
getch();
return 0;
}
```

OUTPUT:

```
Enter the degree of polynomial 1: 2

Enter the degree of polynomial 2: 2

For polynomial 1
Enter the coefficient for exponent 0 : 2

Enter the coefficient for exponent 1 : 3

Enter the coefficient for exponent  2 : 4

For polynomial 2
Enter the coefficient for exponent 0 : 3

Enter the coefficient for exponent 1 : 4

Enter the coefficient for exponent 2:  2

Polynomial  1: 4x^2+3x^1+2
Polynomial 2: 2x^2+4x^1+3

RESULT

6x^2+7x^1+5
```

2. Sequential and binary search : Print number of comparison in each case for given datasets.

```
#include<iostream.h>
#include<conio.h>
#include<process.h>
class search
{
int a[10],i,n,j,k,low,high,mid;
public:
  void get();
  void sort();
  void binarySearch(int a[], int low, int high, int k);
  void getk();
};

void search::get()
{
cout<<"\n enter the size of array:";
cin>>n;
cout<<"\nenter the array elements:";
for(i=0;i<n;i++)
 {
    cin>>a[i];
 }
}

void search::getk()
{
cout<<"\nenter the key to be searched:";
cin>>k;
low=0;
high=n-1;
binarySearch(a,low,high,k);
}

void search::sort()
{
int t;
for(i=0;i<n;i++)
 {
  for(j=i+1;j<n;j++)
   {
    if(a[i]>a[j])
     {
      t=a[i];
      a[i]=a[j];
      a[j]=t;
     }
   }
 }
}

void search::binarySearch(int a[],int low, int high, int k)
```

```
{
if(low>high)
 {
   cout<<"key not fount";
   getch();
   exit(0);
 }
mid=(low+high)/2;
if(k==a[mid])
cout<<"key is found at position"<<mid<<"in sorted array";
else if(k<a[mid])
binarySearch(a,low,mid-1,k);
else if(k>a[mid])
binarySearch(a,mid+1,high,k);
}

int main()
{
clrscr();
search o;
o.get();
o.sort();
o.getk();
getch();
return 0;
}
```

OUTPUT:

```
enter the size of array: 5

enter the array elements: 4  6  7  1  0

enter the key to be search: 7

key is fount at position 4 in sorted array
```

3. Insertion sort: number of comparisons and exchanges for given data sets.

```
#include<iostream.h>
#include<conio.h>
class insertion
{
int i,n,k,a[10],pos;
public:
void get();
void sort();
void display();
};
void insertion::get()
{
cout<<"\nEnter the size:";
cin>>n;
cout<<"\nEnter the array:";
for(i=1;i<=n;i++)
{
cin>>a[i];
}
}
void insertion::sort()
{
for(i=2;i<=n;i++)
{
k=a[i];
pos=i;
while(pos>1&&a[pos-1]>k)
{
a[pos]=a[pos-1];
pos=pos-1;
a[pos]=k;
}
}
}
void insertion::display()
{
cout<<"\nSorted array:";
for(i=1;i<=n;i++)
{
cout<<"\n"<<a[i];
}
}
void main()
{
clrscr();
insertion in;
in.get();
in.sort();
in.display();
getch();
}
```

<u>OUTPUT:</u>

```
Enter the size: 4

Enter the array:

3

2

5

1

Sorted array:

1

2

3

5
```

4. Bubble sort: Print number of comparisons and exchanges for given data sets.

```
#include<iostream.h>
#include<conio.h>
class sort
{
int a[20],i,n,j,t;
public:
void read();
void bubsort();
void display();
};
void sort::read()
{
cout<<"\nenter the number of elements:";
cin>>n;
cout<<"\nenter the elements:";
for(i=1;i<=n;i++)
{
cin>>a[i];
}
}
void sort::bubsort()
{
for(i=1;i<=n-1;i++)
{
for(j=1;j<=n-i;j++)
{
if(a[j]>a[j+1])
{
t=a[j];
a[j]=a[j+1];
a[j+1]=t;
}
}
}
}
void sort::display()
{
cout<<"\nthe sorted array is:";
for(i=1;i<=n;i++)
{
cout<<a[i]<<"\t";
}
}
int main()
{
clrscr();
sort o;
o.read();
o.bubsort();
o.display();
getch();
return 0;
}
```

OUTPUT:

```
enter the number of elements: 5
enter the elements: 9 4 6 3 7
the sorted array is: 3 4 6 7 9
```

5. Selection sort: Print number of comparisons and exchanges for given data sets .

```
#include<iostream.h>
#include<conio.h>
class sort
{
int i,j,n,a[20];
public:
void read();
void selectsort();
void display();
int minimum(int i);
};
void sort::read()
{
cout<<"\nEnter number of elements:";
cin>>n;
cout<<"\nEnter the elements:\n";
for(i=1;i<=n;i++)
{
cin>>a[i];
}
}
void sort::selectsort()
{
for(i=1;i<=n;i++)
{
int min_index=minimum(i);
int temp=a[i];
a[i]=a[min_index];
a[min_index]=temp;
}
}
int sort::minimum(int i)
{
int min_index=i;
for(j=i+1;j<=n;j++)
{
if(a[j]<a[min_index])
min_index=j;
}
return min_index;
}
void sort::display()
{
cout<<"\n Resultant array:\n";
for(i=1;i<=n;i++)
{
cout<<a[i]<<"\t";
}
}
void main()
{
clrscr();
sort s;
s.read();
s.selectsort();
s.display();
getch();
}
```

OUTPUT:

```
Enter the number of elements: 5

Enter the elements: 2  5  8  1  9

Resultant array:  1    2    5    8    9
```

6. Quick sort.

```
#include<iostream.h>
#include<conio.h>
class quicksort
{
int a[10],l,u,i,j,limit;
public:
void read();
void quick(int l,int u);
void display();
};
void quicksort::read()
{
cout<<"\nEnter the limit:";
cin>>limit;
cout<<"\nEnter the elements:";
for(i=0;i<limit;i++)
{
cin>>a[i];
}
i=0;
u=limit-1;
}
void quicksort::display()
{
cout<<"\nElements are...\n";
for(i=0;i<limit;i++)
{
cout<<"\n"<<a[i];
}
quick(l,u);
cout<<"\nSorted elements are...\n";
for(i=0;i<limit;i++)
{
cout<<"\n"<<a[i];
}
}
void quicksort::quick(int l,int u)
{
int p,temp;
if(l<u)
{
p=a[l];
i=l;
j=u;
while(i<j)
{
while(a[i]<=p&&i<j)
i++;
while(a[j]>p&&i<=j)
j--;
if(i<=j)
{
```

```
temp=a[i];
a[i]=a[j];
a[j]=temp;
}
}
temp=a[j];
a[j]=a[l];
a[l]=temp;
quick(l,j-1);
quick(j+1,u);
}
}
void main()
{
clrscr();
quicksort ob;
ob.read();
ob.display();
getch();
}
```

OUTPUT:

```
Enter the limit: 5
Enter the elements: 2 4 5 3 1
Elements are…
2
4
5
3
1
Sorted elements are…
1
2
3
4
5
```

7. Stack operation: addition and deletion of elements

```
#include<iostream.h>
#include<conio.h>
int stack[100],n=100,top=-1;
void push(int val)
{
 if(top>=n-1)
 cout<<"stack overflow"<<endl;
 else
 {
  top++;
  stack[top]=val;
 }
}
void pop()
{
 if(top<=-1)
 {
 cout<<"stack underflow "<<endl;
 }
 else
 {
 cout<<"the popped element is"<<stack[top]<<endl;
 top--;
 }
}
void display()
{
if(top>=0)
{
cout<<"stack elements are:\n";
for(int i=top;i>=0;i--)
{
cout<<stack[i]<<" ";
cout<<endl;
}
}
else
{
cout<<"stack is empty";
}
}
int main()
{
int ch,val;
cout<<"1.push in stack"<<endl;
cout<<"2.pop from stack"<<endl;
cout<<"3.display stack"<<endl;
cout<<"4.exist"<<endl;
do
{
cout<<"Enter your choice:"<<endl;
cin>>ch;
switch(ch)
{
case 1:
cout<<"Enter value to be pushed:"<<endl;
cin>>val;
push(val);
break;
case 2:
pop();
break;
case 3:
display();
break;
case 4:
cout<<"exist:"<<endl;
}
}
while(ch!=4);
return 0;
}
```

OUTPUT:

```
1.push in stack
2.pop in stack
3.display stack
4.exit
Enter your choice:
1
Enter the value to be pushed:
3
Enter your choice:
1
Enter value to be pushed:
5
Enter your choice:
1
Enter value to be pushed:
6
Enter your choice:
3
Stack elements are:
6
5
3
Enter your choice:
2
the popped element is 6
Enter your choice:
3
Stack elements are:
5
3
Enter your choice: 4
```

8. Queue operation: addition and deletion of elements

```
#include<iostream.h>
#include<conio.h>
int queue[50];
int n;
int front=0;
int rear=0;
void get()
{
cout<<"Enter size of queue:";
cin>>n;
}
void Queue_insertion()
{
int val;
if(rear==n)
cout<<"Queue overflow"<<endl;
else
{
cout<<"Insert value in the queue is:\n";
cin>>val;
rear++;
queue[rear]=val;
}
}
void Delete()
{
if(front==rear)
{
cout<<"Queue underflow";
return;
}
else
{
front++;
cout<<"Element delete from queue is:"<<queue[front]<<endl;
}
}
void Display_Queue()
{
if(front==rear)
cout<<"Queue is empty \n";
else
{
cout<<"Queue elements are:";
for(int i=front+1;i<=rear;i++)
cout<<queue[i]<<" ";
cout<<endl;
}
}
int main()
{
int ch;
get();
cout<<"1.insertion element to queue"<<endl;
cout<<"2.delete element from queue"<<endl;
cout<<"3.display all the elements of the queue"<<endl;
cout<<"4.exit"<<endl;
do
{
cout<<"Enter your choice:"<<endl;
cin>>ch;
switch(ch)
{
case 1:Queue_insertion();
break;
case 2:Delete();
break;
case 3:Display_Queue();
break;
case 4:cout<<"Exit"<<endl;
break;
default:cout<<"Invalid choice"<<endl;
}
}
while(ch!=4);
return 0;
}
```

OUTPUT:

```
Enter size of queue: 4
1.insertion element to queue
2.delete element to queue
3.display all the elements of the queue
4.exit
Enter your choice:
1
Insertion value in the queue is:
3
Enter your choice:
1
Insertion value in the queue is:
5
Enter your choice:
1
Insertion value in the queue is:
8
Enter your choice:
```

```
3
Queue elements are: 3   5   8
Enter your choice:
2
Element delete from queue is: 3
Enter your choice:
2
Element delete from queue is: 5
Enter your choice
3
Queue element are: 8
Enter your choice:
4
```

9. Conversion of infix expression to postfix.

```
#include<iostream.h>
#include<string.h>
#include<ctype.h>
#include<conio.h>
const int MAX=50;
class infix
{
private:
char target[MAX],stack[MAX];
char *s,*t;
int top;
public:
infix();
void setexpr(char*str);
void push(char c);
char pop();
void convert();
int priority(char c);
void show();
};
infix::infix()
{
top=-1;
t=target;
s="";
}
void infix::setexpr(char*str)
{
s=str;
}
void infix::push(char c)
{
if(top==MAX)
cout<<"\n stack is full\n";
else
{
top++;
stack[top]=c;
}
}
char infix::pop()
{
if(top==-1)
{
cout<<"\n stack is empty\n";
return -1;
}
else
{
char item=stack[top];
top--;
return item;
```

```
}
}
void infix::convert()
{
while(*s)
{
if(*s==' '||*s=='\t')
{
s++;
continue;
}
if(isdigit(*s)||isalpha(*s))
{
while(isdigit(*s)||isalpha(*s))
{
*t=*s;
s++;
t++;
}
}
if(*s=='(')
{
push(*s);
s++;
}
char opr;
if(*s=='*'||*s=='+'||*s=='/'||*s=='%'||*s=='-'||*s=='$')
{
if(top!=-1)
{
opr=pop();
while(priority(opr)>=priority(*s))
{
*t=opr;
t++;
opr=pop();
}
push(opr);
push(*s);
}
else
push(*s);
s++;
}
if(*s==')')
{
opr=pop();
while((opr)!='(')
{
*t=opr;
t++;
opr=pop();
}
s++;
}
}
while(top!=-1)
{
char opr=pop();
*t=opr;
t++;
}
*t='\0';
}
int infix::priority(char c)
{
if(c=='$')
return 3;
if(c=='*'||c=='/'||c=='%')
return 2;
else
{
if(c=='+'||c=='-')
return 1;
else
return 0;
}
}
void infix::show()
{
cout<<target;
}
void main()
{
clrscr();
char expr[MAX];
infix q;
cout<<"\n enter an expression in infix form:";
cin.getline(expr,MAX);
q.setexpr(expr);
q.convert();
cout<<"\n the postfix expression is:";
q.show();
getch();
}
```

<u>OUTPUT:</u>

```
enter an expression in infix form: A+(B*C)/D

the postfix expression is:ABC*D/+
```

10. Menu driven program: to add / delete elements to a circular queue. Include necessary error messages.

```
#include<iostream.h>
#include<conio.h>
class cq
{
int cq[50],i,l,front,rear,item,n;
public:
 void insert();
 void clear();
 cq()
 {
 front=0;
 rear=0;
 }
void get();
void show();
};

void cq::get()
{
cout<<"\n enter the limit:";
cin>>l;
n=l+1;
}

void cq::insert()
{
int p=rear;
rear=(rear+1)%n;
if(front==rear)
 {
 cout<<"\n circular queue is full";
 rear=p;
 }
else
 {
 cout<<"enter the data:";
 cin>>item;
 cq[rear]=item;
 }
}

void cq::clear()
{
if(front==rear)
 cout<<"\ncircular queue is empty";
else
 {
 front=(front+1)%n;
 item=cq[rear];
 cout<<"\n removed item"<<cq[front];
 }
}
```

```
void cq::show()
{
if(front==rear)
 cout<<"\n queue is empty";
else if(front<rear)
{
 for(i=front+1;i<n;i++)
 {
  cout<<cq[i]<<"\n";
 }
 for(i=0;i<rear;i++)
 {
  cout<<cq[i]<<"\n";
 }
}
}
void main()
{
clrscr();
cq q;
int c;
q.get();
do
{
cout<<"\nmenu:\n 1.push \n 2.pop \n 3.peep \n 4.exit";
cout<<"\n enter your choice(1,2,3,4):";
cin>>c;
switch(c)
 {
 case 1:q.insert();
  break;
 case 2:q.clear();
  break;
 case 3:q.show();
 }
}
while(c<=3);
getch();
}
```

OUTPUT:

```
enter the limit: 3

menu:
1.push
2.pop
3.peep
4.exit
enter your choice(1,2,3,4): 1
enter the data: 2

menu:
```

```
1.push
2.pop
3.peep
4.exit
enter your choice(1,2,3,4): 1
enter the data: 3

menu:
1.push
2.pop
3.peep
4.exit
enter your choice(1,2,3,4): 1
enter the data: 5

menu:
1.push
2.pop
3.peep
4.exit
enter your choice(1,2,3,4): 3
2
3
5

menu:
1.push
2.pop
3.peep
4.exit
enter your choice(1,2,3,4): 2
removed item  2
menu:
1.push
2.pop
3.peep
4.exit
enter your choice(1,2,3,4): 3
3
5

menu:
1.push
2.pop
3.peep
4.exit
enter your choice(1,2,3,4): 4
```

11. Singly linked list operations : add a new node at the beginning, at the end, after i th node, delete from beginning, end, print the list.

```
#include<iostream.h>
#include<conio.h>
#include<process.h>
class linklist
{
struct node
 {
 int data;
 node *link;
 }
*head;
int i;
public:
linklist()
 {
  head=NULL;
 }
void append(int num);
void addbeg(int num);
void addafter(int loc,int num);
void del(int num);
void display();
};

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

void linklist::addbeg(int num)
{
 node *temp;
 temp=new node;
 temp->data=num;
 temp->link=head;
 head=temp;
}

void linklist::addafter(int loc,int num)
{
node *temp,*r;
temp=head;
for(i=1;i<loc;i++)
 {
  temp=temp->link;
  if(temp==NULL)
   {
    cout<<"\n therebarebless than"<<loc<<"elements in list";
    return;
   }
 }
r=new node;
r->data=num;
r->link=temp->link;
temp->link=r;
}

void linklist::del(int num)
{
node *old,*temp;
temp=head;
while(temp!=NULL)
{
 if(temp->data==num)
 {
  if(temp==head)
  {
   head=temp->link;
   delete temp;
  }
  else
  {
   old->link=temp->link;
  }
  delete temp;
  return;
 }
 else
 {
  old=temp;
  temp=temp->link;
 }
}
cout<<"\n\nElement"<<num<<"not found\n";
}

void linklist::display()
```

```
{
 node *temp=head;
 cout<<"\n";
 while(temp!=NULL)
 {
  cout<<temp->data<<" ";
  temp=temp->link;
 }
}

void main()
{
 clrscr();
 linklist l;
 int ch,n,pos;
 do
 {
  cout<<"\n1.Append \n 2.addafter \n 3.add beg \n 4.Delete \n 5.Display \n 6.Exit";
  cout<<"\n enter your choice\n";
  cin>>ch;
  switch(ch)
  {
   case 1:cout<<"\nenter the elements:";
              cin>>n;
              l.append(n);
              break;
   case 2:cout<<"\nenter element to be added and after the position:";
              cin>>n>>pos;
              l.addafter(pos,n);
              break;
   case 3:cout<<"\nenter the element: ";
              cin>>n;
              l.addbeg(n);
              break;
   case 4:cout<<"\nenter the element to be deleted:";
              cin>>n;
              l.del(n);
              break;
   case 5:cout<<"\nelements in the linked list:";
              l.display();
              break;
   case 6:exit(0);
 }
}
while(ch<=5);
getch();
}
```

OUTPUT:

```
1.Append
2.add after
3.add beg
4.Delete
5.Display
```

```
6.Exit
enter your choice: 1
enter the element: 2

1.Append
2.add after
3.add beg
4.Delete
5.Display
6.Exit
enter your choice: 1
enter the element: 3

1.Append
2.add after
3.add beg
4.Delete
5.Display
6.Exit
enter your choice: 5
elements in the linked list:
2 3

1.Append
2.add after
3.add beg
4.Delete
5.Display
6.Exit
enter your choice: 3
enter the element: 5

1.Append
2.add after
3.add beg
4.Delete
5.Display
6.Exit
enter your choice: 5
elements in the linked list:
5  2  3

1.Append
2.add after
3.add beg
4.Delete
5.Display
6.Exit
enter your choice: 2
enter element to be added and after the position: 2 3

1.Append
2.add after
3.add beg
4.Delete
```

```
5.Display
6.Exit
enter your choice: 5
elements in the linked list:
5  2  3  2

1.Append
2.add after
3.add beg
4.Delete
5.Display
6.Exit
enter your choice: 4
enter the element to be deleted: 3

1.Append
2.add after
3.add beg
4.Delete
5.Display
6.Exit
enter your choice: 5
elements in the linked list:
5  2  2

1.Append
2.add after
3.add beg
4.Delete
5.Display
6.Exit
enter your choice: 6
```

12. Circular linked list : add a new node at the beginning, at the end, after ith node, delete from beginning, end, print the list.

```
#include<iostream.h>
#include<stdio.h>
#include<stdlib.h>
struct node
{
int info;
struct node *next;
}
*last;
class circular_llist
{
public:
void create_node(int value);
void add_begin(int value);
void add_after(int value,int position);
void delete_element(int value);
void display_list();
circular_llist()
{
last==NULL;
}
};
void main()
{
int choice,element,position;
circular_llist cl;
while(1)
{
cout<<"\n1.Create node"<<endl;
cout<<"2.Add at beginning"<<endl;
cout<<"3.Add after"<<endl;
cout<<"4.Delete"<<endl;
cout<<"5.Display"<<endl;
cout<<"6.Quit"<<endl;
cout<<"Enter your choice:"<<endl;
cin>>choice;
switch(choice)
{
case 1:
cout<<"Enter the element:";
cin>>element;
cl.create_node(element);
cout<<endl;
break;
case 2:
cout<<"Enter the element:";
cin>>element;
cl.add_begin(element);
cout<<endl;
break;
case 3:
cout<<"Enter the element:";
cin>>element;
cout<<"Insert element after position";
cin>>position;
cl.add_after(element,position);
cout<<endl;
break;
case 4:
if(last==NULL)
{
cout<<"List is empty,nohing to delete"<<endl;
break;
}
cout<<"Enter the element for delete:";
cin>>element;
cl.delete_element(element);
cout<<endl;
break;
case 5:
cl.display_list();
break;
case 6:
exit(1);
break;
default:
cout<<"wrong choice"<<endl;
}
}
}
void circular_llist::create_node(int value)
{
struct node *temp;
temp=new(struct node);
temp->info=value;
if(last==NULL)
{
last=temp;
temp->next=last;
}
else
{
temp->next=last->next;
last->next=temp;
last=temp;
}
}
void circular_llist::add_begin(int value)
{
if(last==NULL)
{
cout<<"first create the list"<<endl;
return;
}
struct node *temp;
temp=new(struct node);
temp->info=value;
```

```
temp->next=last->next;
last->next=temp;
}
void circular_llist::add_after(int value,int pos)
{
if(last==NULL)
{
cout<<"first create the list"<<endl;
return;
}
struct node *temp,*s;
s=last->next;
for(int i=0;i<pos-1;i++)
{
s=s->next;
if(s==last->next)
{
cout<<"there are less than";
cout<<"pos"<< "in the list"<<endl;
return;
}
}
temp=new(struct node);
temp->next=s->next;
temp->info=value;
s->next=temp;
if(s==last)
{
last=temp;
}
}
void circular_llist::delete_element(int value)
{
struct node *temp,*s;
s=last->next;
if(last->next==last&&last->info==value)
{
temp=last;
last=NULL;
free(temp);
return;
}
if(s->info==value)
{
temp=s;
last->next=s->next;
free(temp);
return;
}
while(s->next!=last)
{
if(s->next->info==value)
{
temp=s->next;
s->next=temp->next;
free(temp);
cout<<"element"<<value;
cout<<"deleted from the list"<<endl;
return;
}
s=s->next;
}
if(s->next->info==value)
{
temp=s->next;
s->next=last->next;
free(temp);
last=s;
return;
}
cout<<"Element"<<value<< "not found in the list"<<endl;
}

void circular_llist::display_list()
{
struct node *s;
if(last==NULL)
{
cout<<"List is empty,nothing to display"<<endl;
return;
}
s=last->next;
cout<<"circular link list:"<<endl;
while(s!=last)
{
cout<<s->info<<"->";
s=s->next;
}
cout<<s->info<<endl;
}
```

OUTPUT:

```
1.Create node
2.Add at beginning
3.Add after
4.Delete
5.Display
6.Quit
Enter your choice:
1
Enter the element:2

 1.Create node
2.Add at beginning
3.Add after
4.Delete
5.Display
6.Quit
Enter your choice:
```

```
1
Enter the element: 5

1.Create node
2.Add at beginning
3.Add after
4.Delete
5.Display
6.Quit
Enter your choice:
5
Circular link list:
2->5

1.Create node
2.Add at beginning
3.Add after
4.Delete
5.Display
6.Quit
Enter your choice:
2
Enter the element:6

1.Create node
2.Add at beginning
3.Add after
4.Delete
5.Display
6.Quit
Enter your choice:
5
Circular link list:
6->2->5

1.Create node
2.Add at beginning
3.Add after
4.Delete
5.Display
6.Quit
Enter your choice:
3
Enter the element:1
Insert element after position 2

1.Create node
2.Add at beginning
3.Add after
4.Delete
5.Display
6.Quit
Enter your choice:
5
```

```
Circular link list:
6->2->1->5

1.Create node
2.Add at beginning
3.Add after
4.Delete
5.Display
6.Quit
Enter your choice:
4
Enter the element for delete: 5

1.Create node
2.Add at beginning
3.Add after
4.Delete
5.Display
6.Quit
Enter your choice:
5
Circular link list:
6->2->1

1.Create node
2.Add at beginning
3.Add after
4.Delete
5.Display
6.Quit
Enter your choice:
6
```

13. Doubly linked list : add a new node at the beginning, at the end, after ith node, delete from beginning, end, print the list.

```
#include<iostream.h>
#include<stdio.h>
#include<stdlib.h>
struct node
{
int info;
struct node *next;
struct node *prev;
}*start;
class double_llist
{
public:
void create_list(int value);
void add_begin(int value);
void add_after(int value,int position);
void delete_element(int value);
void display_dlist();
double_llist()
{
start=NULL;
}
};
void main()
{
int choice, element, position;
double_llist dl;
while(1)
{
cout<<"\n1.Create node"<<endl;
cout<<"2.Add at beginning"<<endl;
cout<<"3.Add after position"<<endl;
cout<<"4.delete"<<endl;
cout<<"5.Display"<<endl;
cout<<"6.Quit"<<endl;
cout<<"Enter your choice:";
cin>>choice;
switch(choice)
{
case 1:
cout<<"Enter the element:";
cin>>element;
dl.create_list(element);
cout<<endl;
break;
case 2:
cout<<"Enter the element:";
cin>>element;
dl.add_begin(element);
cout<<endl;
break;
case 3:
cout<<"Enter the element:";
cin>>element;
cout<<"Insert element after position:";
cin>>position;
dl.add_after(element,position);
cout<<endl;
break;
case 4:
if(start==NULL)
{
cout<<"List empty,nothing to delete"<<endl;
break;
}
cout<<"Enter element for deletion:";
cin>>element;
dl.delete_element(element);
cout<<endl;
break;
case 5:
dl.display_dlist();
cout<<endl;
break;
case 6:
exit(1);
default:
cout<<"Wrong choice"<<endl;
}
}
}

void double_llist::create_list(int value)
{
struct node *s,*temp;
temp=new (struct node);
temp->info=value;
temp->next=NULL;
if(start==NULL)
{
temp->prev=NULL;
start=temp;
}
else
{
s=start;
while(s->next!=NULL)
s=s->next;
s->next=temp;
temp->prev=s;
}
}

void double_llist::add_begin(int value)
{
if(start==NULL)
{
cout<<"First create the list"<<endl;
```

```
return;
}
struct node *temp;
temp=new(struct node);
temp->prev=NULL;
temp->info=value;
temp->next=start;
start->prev=temp;
start=temp;
cout<<"element inserted"<<endl;
}

void double_llist::add_after(int value,int pos)
{
if(start==NULL)
{
cout<<"First create the list"<<endl;
return;
}
struct node *tmp,*q;
int i;
q=start;
for(i=0;i<pos-1;i++)
{
q=q->next;
if(q==NULL)
{
cout<<"There are less than";
cout<<pos<<"elements"<<endl;
return;
}
}
tmp=new(struct node);
tmp->info=value;
if(q->next==NULL)
{
q->next=tmp;
tmp->next=NULL;
tmp->prev=q;
}
else
{
tmp->next=q->next;
tmp->next->prev=tmp;
q->next=tmp;
tmp->prev=q;
}
cout<<"Element inserted"<<endl;
}

void double_llist::delete_element(int value)
{
struct node *tmp,*q;
if(start->info==value)
{
```

```
tmp=start;
start=start->next;
start->prev=NULL;
cout<<"element deleted"<<endl;
free(tmp);
return;
}
q=start;
while(q->next->next!=NULL)
{
if(q->next->info==value)
{
tmp=q->next;
q->next=tmp->next;
tmp->next->prev=q;
cout<<"element deleted"<<endl;
free(tmp);
return;
}
q=q->next;
}
if(q->next->info==value)
{
tmp=q->next;
free(tmp);
q->next=NULL;
cout<<"element deleted"<<endl;
return;
}
cout<<"element"<<value<<" not found"<<endl;
}

void double_llist::display_dlist()
{
struct node *q;
if(start==NULL)
{
cout<<"list empty,nothing to display"<<endl;
return;
}
q=start;
cout<<"the doubly linked list is:"<<endl;
while(q!=NULL)
{
cout<<q->info<<"<->";
q=q->next;
}
cout<<"NULL"<<endl;
}
```

OUTPUT:

```
1.Create node
2.Add at beginning
3.Add after position
```

```
4.delete
5.Display
6.Quit
Enter your choice: 1
Enter the element: 6

1.Create node
2.Add at beginning
3.Add after position
4.delete
5.Display
6.Quit
Enter your choice: 1
Enter the element: 7

1.Create node
2.Add at beginning
3.Add after position
4.delete
5.Display
6.Quit
Enter your choice: 5
the doubly list is:
6<->7<->NULL

1.Create node
2.Add at beginning
3.Add after position
4.delete
5.Display
6.Quit
Enter your choice: 2
Enter the element: 4
element inserted

1.Create node
2.Add at beginning
3.Add after position
4.delete
5.Display
6.Quit
Enter your choice: 5
The doubly linked list is:
4<->6<->7<->NULL

1.Create node
2.Add at beginning
3.Add after position
4.delete
5.Display
6.Quit
```

```
Enter your choice: 3
Enter the element: 8
Insert element after position: 3
element inserted

1.Create node
2.Add at beginning
3.Add after position
4.delete
5.Display
6.Quit
Enter your choice:5
The doubly linked list is:
4<->6<->7<->8<->NULL

1.Create node
2.Add at beginning
3.Add after position
4.delete
5.Display
6.Quit
Enter your choice: 4
Enter element for deletion: 6
element deleted

1.Create node
2.Add at beginning
3.Add after position
4.delete
5.Display
6.Quit
Enter your choice:6
```

14. Implement tree traversal.

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
struct btree
{
struct btree *left;
struct btree *right;
int no;
};
void postorder(struct btree *trav);
void inorder(struct btree *trav);
void preorder(struct btree *trav);
struct btree *create(struct btree *trav);
void main()
{
struct btree *root=NULL;
char c;
clrscr();
while(1)
{
root=create(root);
cout<<"\nDo you want to continue(y/n):";
cin>>c;
if(c=='n'||c=='N')
break;
}
cout<<"\nInorder is:";inorder(root);
cout<<"\nPreorder is:";preorder(root);
cout<<"\nPostorder is:";postorder(root);
getch();
}
struct btree *create(struct btree *trav)
{
if(trav==NULL)
{
trav=new btree;
trav->right=NULL;
trav->left=NULL;
cout<<"\nEnter the data:";
cin>>trav->no;
return trav;
}
else
{
char choice;
cout<<"\nDo you want to create left or right child(L for left R for right):";
cin>>choice;
if(choice=='r'||choice=='R')
{
trav->right=create(trav->right);
}
if(choice=='l'||choice=='L')
trav->left=create(trav->left);
}
return trav;
}
void inorder(struct btree *trav)
{
if(trav==NULL)
return;
inorder(trav->left);
cout<<trav->no<<" ";
inorder(trav->right);
}
void preorder(struct btree *trav)
{
if(trav==NULL)
return;
cout<<trav->no<<" ";
preorder(trav->left);
preorder(trav->right);
}
void postorder(struct btree *trav)
{
if(trav==NULL)
return;
postorder(trav->left);
postorder(trav->right);
cout<<trav->no<<" ";
}
```

OUTPUT:

```
Enter the data:4

Do you want to continue(y/n):y

Do you want to create left or right child(L for left R for right):R

Enter the data:1

Do you want to continue(y/n):y

Do you want to create left or right child(L for left R for right):L

Enter the data:2

Do you want to continue(y/n):y

Do you want to create left or right child(L for left R for right):R

Do you want to create left or right child(L for left R for right):L

Enter the data:6

Do you want to continue(y/n):n

Inorder is:2  4  6  1
Preorder is:4  2  1  6
Postorder is:2  6  1  4
```

15. Merge two sorted linked list.

```
#include<iostream.h>
#include<conio.h>
struct node
{
int data;
struct node *next;
};
class list
{
struct node *start;
public:
void create();
void show();
void merge(list,list);
};
void list::create()
{
struct node *nxt_node,*pre_node;
int value,no,i;
start=nxt_node=pre_node=NULL;
cout<<"\nHow many nodes:";
cin>>no;
cout<<"Enter the "<<no<<" elements:\n";
for(i=1;i<=no;i++)
{
cin>>value;
nxt_node=new node;
nxt_node->data=value;
nxt_node->next=NULL;
if(start==NULL)
start=nxt_node;
else
pre_node->next=nxt_node;
pre_node=nxt_node;
}
cout<<"\nThe list is created\n";
}
void list::show()
{
struct node *ptr=start;
while(ptr!=NULL)
{
cout<<ptr->data<<" ";
ptr=ptr->next;
}
cout<<" ";
}
void list::merge(list l1,list l2)
{
```

```
struct node *nxt_node,*pre_node,*pptr,*qptr;
int dat;
pptr=l1.start;
qptr=l2.start;
start=nxt_node=pre_node=NULL;
while(pptr!=NULL&&qptr!=NULL)
{
if(pptr->data<=qptr->data)
{
dat=pptr->data;
pptr=pptr->next;
}
else
{
dat=qptr->data;
qptr=qptr->next;
}
nxt_node=new node;
nxt_node->data=dat;
nxt_node->next=NULL;
if(start==NULL)
start=nxt_node;
else
pre_node->next=nxt_node;
pre_node=nxt_node;
}
if(pptr==NULL)
{
while(qptr!=NULL)
{
nxt_node=new node;
nxt_node->data=qptr->data;
nxt_node->next=NULL;
if(start==NULL)
start=nxt_node;
else
pre_node->next=nxt_node;
pre_node=nxt_node;
qptr=qptr->next;
}
}
else if(qptr==NULL)
{
while(pptr!=NULL)
{
nxt_node=new node;
nxt_node->data=pptr->data;
nxt_node->next=NULL;
if(start==NULL)
start=nxt_node;
else
pre_node->next=nxt_node;
pre_node=nxt_node;
pptr=pptr->next;
}
}
cout<<"\nThe lists are merged\n";
return;
}
int main()
{
clrscr();
list l1,l2,l3;
cout<<"\nEnter the list in ascending order:\n";
l1.create();
cout<<"\nEnter the second list in ascending order:\n";
l2.create();
cout<<"The first list is:\n";
l1.show();
cout<<"\nThe second list is:\n";
l2.show();
l3.merge(l1,l2);
l3.show();
getch();
return 0;
}
```

OUTPUT:

```
Enter the list in ascending order:

How many nodes:5
Enter the 5 elements:
1
3
5
7
9

The list is created

Enter the second list in ascending order:

How many nodes:5
Enter the 5 elements:
2
4
6
8
9
The list is created
The first list is:
1  3  5  7  9
The second list is:
2  4  6  8  9
The lists are merged
1  2  3  4  5  6  7  8  9  9
```

16. Linear Search and Binary Search.

```
#include<iostream.h>
#include<conio.h>
#include<process.h>
class search
{
int a[10],i,n,low,high,mid,j;
public:
void getl();
void linear();
void sort();
void getb();
void binary_search(int a[],int low,int high,int k);
};

void search::getl()
{
cout<<"\nEnter the size:";
cin>>n;
cout<<"\nEnter  the array elements:";
for(i=1;i<=n;i++)
{
cin>>a[i];
}
}

void search::linear()
{
int k;
cout<<"\nEnter the key to be searched:";
cin>>k;
i=0;
while(i<n&&k!=a[i])
{
i++;
}
if(k==a[i])
{
cout<<"\nElement present at:"<<i;
}
else
{
cout<<"\nElement not found";
}
}

void search::sort()
{
int temp;
for(i=1;i<=n;i++)
{
for(j=i+1;j<=n;j++)
{
```

```
if(a[i]>a[j])
{
temp=a[i];
a[i]=a[j];
a[j]=temp;
}
}
}
}

void search::getb()
{
int k;
cout<<"\nEnter the element to be search:";
cin>>k;
low=1;
high=n;
binary_search(a,low,high,k);
}

void search::binary_search(int a[],int low,int high,int k)
{
if(low>high)
{
cout<<"\nKey not found";
getch();
exit(0);
}
mid=(low+high)/2;
if(k==a[mid])
cout<<"\nKey found at position "<<mid<<" in sorted array";
else if(k<a[mid])
binary_search(a,low,mid-1,k);
else if(k>a[mid])
binary_search(a,mid+1,high,k);
}

int main()
{
clrscr();
search s;
int ch;
s.getl();
do
{
cout<<"\nMAIN MENU\n1.Linear search\n2.Binary search\n3.Exit\nEnter your choice:";
cin>>ch;
switch(ch)
{
case 1:
s.linear();
break;
case 2:
s.sort();
s.getb();
```

```
break;
case 3:
break;
}
}
while(ch<3);
getch();
return 0;
}
```

OUTPUT:

```
Enter the size:5

Enter the array elements: 5  3  7  1  9

MAIN MENU
1.Linear search
2.Binary search
3.Exit
Enter your choice:1

Enter the key to be searched:3

Element present at:2

MAIN MENU
1.Linear search
2.Binary search
3.Exit
Enter your choice:2

Enter the element to be search:7

Key found at position 4 in sorted array

MAIN MENU
1.Linear search
2.Binary search
3.Exit
Enter your choice:3
```

SQL:1

Create table students with fields Sno , Sname , Sex ,Mark with Sno as primary key and assign suitable constraints for each attribute . Insert five records into the table.

CREATE TABLE students(Sno int PRIMARY KEY,Sname varchar(20) NOT NULL,Sex varchar(10) NOT NULL,Mark float NOT NULL);

INSERT INTO students VALUES(1001,'Rahul','Male',85.5);

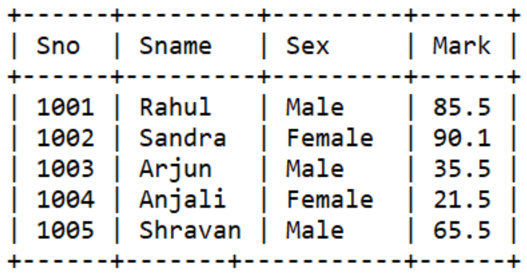
INSERT INTO students VALUES(1002,'Sandra','Female',90.1);

INSERT INTO students VALUES(1003,'Arjun','Male',35.5);

INSERT INTO students VALUES(1004,'Anjali','Female',21.5);

INSERT INTO students VALUES(1005,'Shravan','Male',65.5);

SELECT * FROM students;



```
+------+---------+---------+------+
| Sno  | Sname   | Sex     | Mark |
+------+---------+---------+------+
| 1001 | Rahul   | Male    | 85.5 |
| 1002 | Sandra  | Female  | 90.1 |
| 1003 | Arjun   | Male    | 35.5 |
| 1004 | Anjali  | Female  | 21.5 |
| 1005 | Shravan | Male    | 65.5 |
+------+-------+----------+------+
```

1.Alter the table by adding one more field rank.

ALTER TABLE students ADD Ranks int;

UPDATE students SET Ranks=1 WHERE Sno=1002;

UPDATE students SET Ranks=2 WHERE Sno=1001;

UPDATE students SET Ranks=3 WHERE Sno=1005;

UPDATE students SET Ranks=4 WHERE Sno=1003;

UPDATE students SET Ranks=5 WHERE Sno=1004;

SELECT * FROM students;

```
+------+---------+--------+------+-------+
| Sno  | Sname   | Sex    | Mark | Ranks |
+------+---------+--------+------+-------+
| 1001 | Rahul   | Male   | 85.5 |     2 |
| 1002 | Sandra  | Female | 90.1 |     1 |
| 1003 | Arjun   | Male   | 35.5 |     4 |
| 1004 | Anjali  | Female | 21.5 |     5 |
| 1005 | Shravan | Male   | 65.5 |     3 |
+------+---------+--------+------+-------+
```

2.Display all boys students with their name.

SELECT Sname FROM students WHERE SEX='Male';

```
+-----------+
| Sname     |
+-----------+
| Rahul     |
| Arjun     |
| Shravan   |
+-----------+
```

3.Find the average mark.

SELECT avg(Mark) FROM students;

```
+--------------------+
| avg(Mark)          |
+--------------------+
| 59.619999694824216 |
+--------------------+
```

4.Create a query to display the Sno and Sname for all students who got more than average mark sort the result in the descending order of mark.

SELECT Sno,Sname FROM students WHERE Mark>(SELECT avg(Mark) from students) order by mark desc;

```
+------+---------+
| Sno  | Sname   |
+------+---------+
| 1002 | Sandra  |
| 1001 | Rahul   |
| 1005 | Shravan |
+------+---------+
```

5.Display all girls students those for having marks greater than 20 and less than 40.

SELECT Sname FROM students WHERE Mark BETWEEN 20 AND 40 AND Sex="Female";

```
+--------+
| Sname  |
+--------+
| Anjali |
+--------+
```

SQL:2

Create a table department with fields Ename , Salary , Dno , Dname , Place with Dno as Primary key. Insert five records into the table.

CREATE TABLE department(Dno int PRIMARY KEY,Ename varchar(20),Salary float,Dname varchar(20),Place varchar(20));

INSERT INTO department VALUES(101,'Ramu',10000.50,'Sales','Kannur');

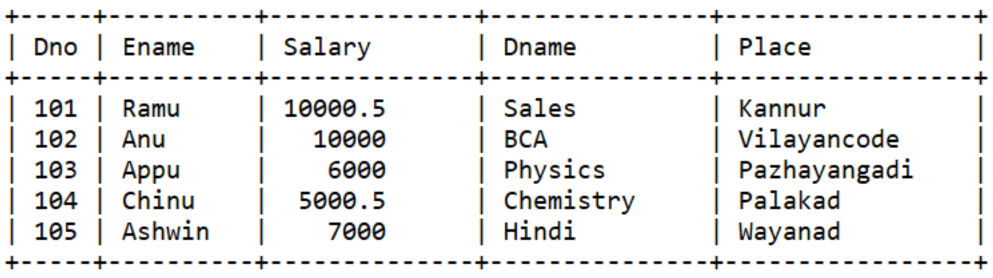
INSERT INTO department VALUES(102,'Anu',10000,'BCA','Vilayancode');

INSERT INTO department VALUES(103,'Appu',6000,'Physics','Pazhayangadi');

INSERT INTO department VALUES(104,'Chinu',5000.5,'Chemistry','Palakad');

INSERT INTO department VALUES(105,'Ashwin',7000,'Hindi','Wayanad');

SELECT * FROM department;

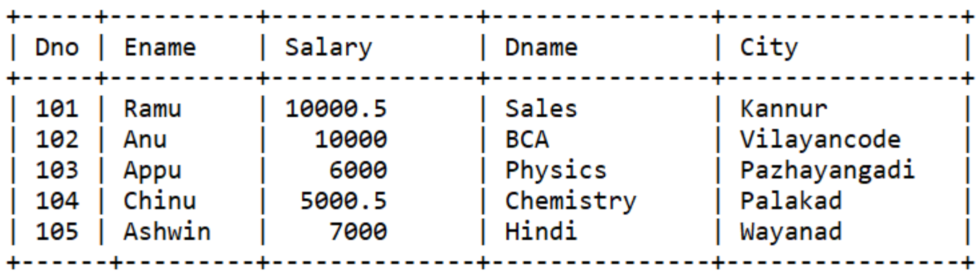


```
+-----+----------+-------------+--------------+-----------------+
| Dno | Ename    | Salary      | Dname        | Place           |
+-----+----------+-------------+--------------+-----------------+
| 101 | Ramu     | 10000.5     | Sales        | Kannur          |
| 102 | Anu      |   10000     | BCA          | Vilayancode     |
| 103 | Appu     |    6000     | Physics      | Pazhayangadi    |
| 104 | Chinu    |  5000.5     | Chemistry    | Palakad         |
| 105 | Ashwin   |    7000     | Hindi        | Wayanad         |
+-----+----------+-------------+--------------+-----------------+
```

1.Rename the field 'Place' with 'City'.

ALTER TABLE department CHANGE Place City varchar(20);

SELECT * FROM department;



```
+-----+----------+-------------+--------------+----------------+
| Dno | Ename    | Salary      | Dname        | City           |
+-----+----------+-------------+--------------+----------------+
| 101 | Ramu     | 10000.5     | Sales        | Kannur         |
| 102 | Anu      |   10000     | BCA          | Vilayancode    |
| 103 | Appu     |    6000     | Physics      | Pazhayangadi   |
| 104 | Chinu    |  5000.5     | Chemistry    | Palakad        |
| 105 | Ashwin   |    7000     | Hindi        | Wayanad        |
+------+---------+-------------+--------------+----------------+
```

2.Display the employees who got salary more than 6000 and less than 10000.

SELECT Ename FROM department WHERE Salary>6000 AND Salary<10000;

```
+--------+
| Ename  |
+--------+
| Ashwin |
+--------+
```

3.Display total salary of organisation.

SELECT sum(Salary) AS TOTAL FROM department;

```
+-------+
| TOTAL |
+-------+
| 38001 |
+-------+
```

4.Display Ename for those who are getting salary in between 5000 and 10000.

SELECT Ename FROM department WHERE Salary BETWEEN 5000 AND 10000;

```
+--------+
| Ename  |
+--------+
| Anu    |
| Appu   |
| Chinu  |
| Ashwin |
+--------+
```

5.Create a view named 'star' with field Ename , Salary and Place.

CREATE VIEW star AS SELECT Ename,Salary,Place FROM department;

SELECT * FROM star;

```
+--------+---------+-------------+
| Ename  | Salary  | Place       |
+--------+---------+-------------+
| Ramu   | 10000.5 | Kannur      |
| Anu    | 10000   | Vilayancode |
| Appu   | 6000    | Pazhayangadi |
| Chinu  | 5000.5  | Palakad     |
| Ashwin | 7000    | Wayanad     |
+--------+---------+-------------+
```

6.Display Ename and Salary with salary rounded with ten digits.

SELECT round(Salary,10) ,Ename FROM department;

```
+------------------------+-------------+
|     Round(Salary,10)   | Ename       |
+------------------------+-------------+
| 10000.5000000000       | Ramu        |
| 10000.0000000000       | Anu         |
| 6000.0000000000        | Appu        |
| 5000.5000000000        | Chinu       |
| 7000.0000000000        | Ashwin      |
+------------------------+-------------+
```

SQL:3

Create a table department with fields dno , dname , dmanager and places 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.

Insert five records into each table.

CREATE TABLE department(Dno int primary key,Dname varchar(20),Dmanager varchar(20),Place varchar(20));

INSERT INTO department VALUES(100,'Production','Priya','Kannur');

INSERT INTO department VALUES(101,'Accounting','Pranav','Kochi');

INSERT INTO department VALUES(102,'Sales','Anu','Wayanad');

INSERT INTO department VALUES(103,'Marketing','Surya','Payyanur');

INSERT INTO department VALUES(104,'Finance','Vishnu','Payyanur');

SELECT * FROM department;

```
+-----+------------+----------+----------+
| Dno | Dname      | Dmanager | Place    |
+-----+------------+----------+----------+
| 100 | Production | Priya    | Kannur   |
| 101 | Accounting | Pranav   | Kochi    |
| 102 | Sales      | Anu      | Wayanad  |
| 103 | Marketing  | Surya    | Payyanur |
| 104 | Finance    | Vishnu   | Payyanur |
+-----+------------+----------+----------+
```

CREATE TABLE emp(Eno int primary key,Ename varchar(20),Job varchar(20),Dno int,foreign key(Dno) references department(Dno),Salary int);

INSERT INTO emp VALUES(10,'Arjun','Wood workers',100,3000);

INSERT INTO emp VALUES(20,'Joseph','Accountant',101,6000);

INSERT INTO emp VALUES(30,'Lisa','Sales man',102,8000);

INSERT INTO emp VALUES(40,'Anil','Executive',103,10000);

INSERT INTO emp VALUES(50,'Elen','Insurance',104,8000);

SELECT * FROM emp;

```
+-----+--------+--------------+------+--------+
| Eno | Ename  | Job          | Dno  | Salary |
+-----+--------+--------------+------+--------+
|  10 | Arjun  | Wood workers |  100 |   3000 |
|  20 | Joseph | Accountant   |  101 |   6000 |
|  30 | Lisa   | Sales man    |  102 |   8000 |
|  40 | Anil   | Executive    |  103 |  10000 |
|  50 | Elen   | Insurance    |  104 |   8000 |
+-----+--------+--------------+------+--------+
```

1.Display the ename and salary with ascending order.

SELECT Ename,Salary FROM emp ORDER BY Salary ASC;

```
+--------+--------+
| Ename  | Salary |
+--------+--------+
| Arjun  |   3000 |
| Joseph |   6000 |
| Lisa   |   8000 |
| Elen   |   8000 |
| Anil   |  10000 |
+--------+--------+
```

2.Display ename and salary for eno=20.

SELECT Ename,Salary FROM emp WHERE Eno=20;

```
+--------+--------+
| Ename  | Salary |
+--------+--------+
| Joseph |   6000 |
+--------+--------+
```

3.Display the manager for the accounting Department.

SELECT Dmanager FROM department WHERE Dname='Accounting';

```
+----------+
| Dmanager |
+----------+
| Pranav   |
+----------+
```

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

SELECT Ename,Salary,Dmanager from emp,department where department.dno=emp.dno AND Salary>5000;

```
+--------+--------+----------+
| Ename  | Salary | Dmanager |
+--------+--------+----------+
| Joseph |   6000 | Pranav   |
| Lisa   |   8000 | Anu      |
| Anil   |  10000 | Surya    |
| Elen   |   8000 | Vishnu   |
+--------+--------+----------+
```

5.Write the queries using various group functions.

SELECT avg(Salary) FROM emp ;

```
+-------------+
| avg(Salary) |
+-------------+
|   7000.0000 |
+-------------+
```

SELECT max(Salary) FROM emp;

```
+-------------+
| max(Salary) |
+-------------+
|       10000 |
+-------------+
```

SELECT min(Salary) FROM emp;

```
+-------------+
| min(Salary) |
+-------------+
|        3000 |
+-------------+
```

SELECT count(Salary) FROM emp;

```
+---------------+
| count(Salary) |
+---------------+
|             5 |
+---------------+
```

SELECT sum(Salary) FROM emp;

```
+-------------+
| sum(Salary) |
+-------------+
|       35000 |
+-------------+
```

SELECT round(Salary) FROM emp;

```
+---------------+
| round(Salary) |
+---------------+
|          3000 |
|          6000 |
|          8000 |
|         10000 |
|          8000 |
+---------------+
```

6.Write the queries using various Number functions.

SELECT greatest(3000,6000,8000,10000,8000);

```
+--------------------------------------+
| greatest( 3000,6000,8000,10000,8000) |
+--------------------------------------+
|                                10000 |
+--------------------------------------+
```

SELECT sqrt(6000);

```
+-------------------+
| sqrt(6000)        |
+-------------------+
| 77.45966692414834 |
+-------------------+
```

SELECT mod(8000,3);

```
+-------------+
| mod(8000,3) |
+-------------+
|           2 |
+-------------+
```

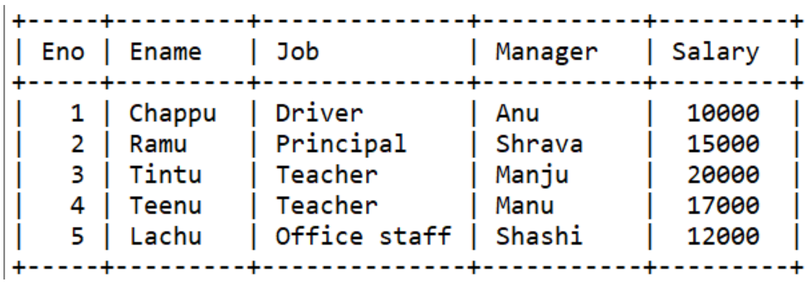
SQL:4

Create a table employee with fields Eno, Ename, Job, Manager and salary, with Eno as primary key. Insert value into the table.

```
CREATE TABLE employee(Eno int primary key,Ename varchar(20),Job varchar(15),Manager varchar(20),Salary float);
```

```
INSERT INTO employee VALUES(1,'Chappu','Driver','Anu',10000);

INSERT INTO employee VALUES(2,'Ramu','Principal','Shrava',15000);

INSERT INTO employee VALUES(3,'Tintu','Teacher','Manju',20000);

INSERT INTO employee VALUES(4,'Teenu','Teacher','Manu',17000);

INSERT INTO employee VALUES(5,'Lachu','Office staff','Shashi',12000);
```

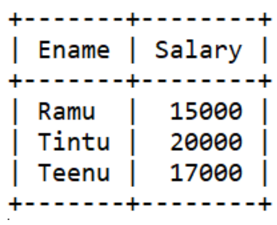
```
SELECT * FROM employee;
```



```
+-----+---------+--------------+-----------+---------+
| Eno | Ename   | Job          | Manager   | Salary  |
+-----+---------+--------------+-----------+---------+
|   1 | Chappu  | Driver       | Anu       |  10000  |
|   2 | Ramu    | Principal    | Shrava    |  15000  |
|   3 | Tintu   | Teacher      | Manju     |  20000  |
|   4 | Teenu   | Teacher      | Manu      |  17000  |
|   5 | Lachu   | Office staff | Shashi    |  12000  |
+-----+---------+--------------+-----------+---------+
```

1.Display Ename, Salary, from employee who are getting salary more than the average salary of the organization.

```
SELECT Ename,Salary FROM employee WHERE Salary>(SELECT avg(Salary) FROM employee);
```



```
+-------+--------+
| Ename | Salary |
+-------+--------+
| Ramu  |  15000 |
| Tintu |  20000 |
| Teenu |  17000 |
+-------+--------+
```

2.Add 20% DA as the extra salary to all employees. Label column as 'New salary'.

```
ALTER TABLE employee ADD New_Salary int;

UPDATE employee SET New_Salary=Salary+(Salary*0.2);
```

SELECT * FROM employee;

```
+-----+--------+--------------+---------+--------+------------+
| Eno | Ename  | Job          | Manager | Salary | New_Salary |
+-----+--------+--------------+---------+--------+------------+
|   1 | Chappu | Driver       | Anu     |  10000 |      12000 |
|   2 | Ramu   | Principal    | Shrava  |  15000 |      18000 |
|   3 | Tintu  | Teacher      | Manju   |  20000 |      24000 |
|   4 | Teenu  | Teacher      | Manu    |  17000 |      20400 |
|   5 | Lachu  | Office staff | Shashi  |  12000 |      14400 |
+-----+--------+--------------+---------+--------+------------+
```

3.Create a query to display the Eno and Ename for all employees who earn more than the average salary.Sort the result in descending order of salary.

SELECT Eno,Ename FROM employee WHERE Salary>(SELECT avg(Salary) FROM employee ORDER BY Salary DESC);

```
+-----+-------+
| Eno | Ename |
+-----+-------+
|   2 | Ramu  |
|   3 | Tintu |
|   4 | Teenu |
+-----+-------+
```

4.Create a view called emp_view based on the Eno, Ename from employee table change the heading of the Ename to 'EMPLOYEE'.

CREATE VIEW emp_view(eno,employee) AS SELECT eno,ename FROM employee;

ALTER VIEW emp_view(eno,employee)as select eno,ename FROM employee;

SELECT * FROM emp_view;

```
+-----+----------+
| eno | employee |
+-----+----------+
|   1 | Chappu   |
|   2 | Ramu     |
|   3 | Tintu    |
|   4 | Teenu    |
|   5 | Lachu    |
+-----+----------+
```

5.Write a query that will display the Eno and Ename for all employees whose name contain 'T'.

SELECT Eno,Ename FROM employee WHERE Ename LIKE '%t%';

```
+------+-------+
| Eno  | Ename |
+------+-------+
|   3  | Tintu |
|   4  | Teenu |
+------+-------+
```

SQL:5

Create a table department with fields dno , ename , salary , designation , dname and place with dno as primary key . Insert values into the table.

CREATE TABLE department(Dno int primary key,Ename varchar(20),Salary int,Designation varchar(20),Dname varchar(20),Place varchar(20));

INSERT INTO department VALUES(101,'Amala',5000,'Clerk','Bcom','Payangadi');

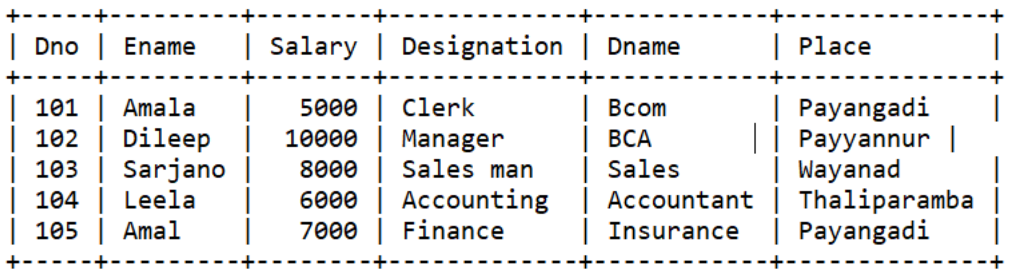
INSERT INTO department VALUES(102,'Dileep',10000,'Manager','BCA','Payyannur');

INSERT INTO department VALUES(103,'Sarjano',8000,'Sales man','Sales','Wayanad');

INSERT INTO department VALUES(104,'Leela',6000,'Accounting','Accountant','Thaliparamba');

INSERT INTO department VALUES(105,'Amal',7000,'Finance','Insurance','Payangadi');

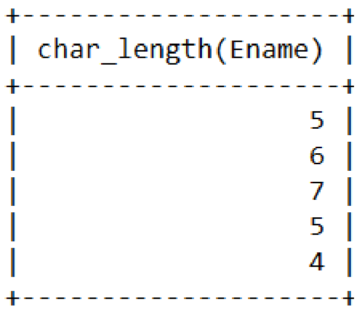
SELECT * FROM department;



```
+-----+---------+--------+-------------+------------+--------------+
| Dno | Ename   | Salary | Designation | Dname      | Place        |
+-----+---------+--------+-------------+------------+--------------+
| 101 | Amala   |   5000 | Clerk       | Bcom       | Payangadi    |
| 102 | Dileep  |  10000 | Manager     | BCA       | | Payyannur |
| 103 | Sarjano |   8000 | Sales man   | Sales      | Wayanad      |
| 104 | Leela   |   6000 | Accounting  | Accountant | Thaliparamba |
| 105 | Amal    |   7000 | Finance     | Insurance  | Payangadi    |
+-----+---------+--------+-------------+------------+--------------+
```

1.Write the queries using various Character functions in ename field.

SELECT char_length(Ename) FROM department;



```
+--------------------+
| char_length(Ename) |
+--------------------+
|                  5 |
|                  6 |
|                  7 |
|                  5 |
|                  4 |
+--------------------+
```

SELECT ucase(Ename)from department;

```
+--------------+
| ucase(Ename) |
+--------------+
| AMALA        |
| DILEEP       |
| SARJANO      |
| LEELA        |
| AMAL         |
+--------------+
```

SELECT lcase(Ename)from department;

```
+--------------+
| lcase(Ename) |
+--------------+
| amala        |
| dileep       |
| sarjano      |
| leela        |
| amal         |
+--------------+
```

SELECT mid(Ename,4,3) FROM department;

```
+----------------+
| mid(Ename,4,3) |
+----------------+
| la             |
| eep            |
| jan            |
| la             |
| l              |
+----------------+
```

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

SELECT Dno,Ename FROM department WHERE Salary>(SELECT avg(Salary) FROM department ORDER BY Salary DESC);

```
+-----+---------+
| Dno | Ename   |
+-----+---------+
| 102 | Dileep  |
| 103 | Sarjano |
+-----+---------+
```

3.Display all employees who got salary between 5000 and 10000.

SELECT Ename FROM department WHERE Salary BETWEEN 5000 AND 10000;

```
+---------+
| Ename   |
+---------+
| Amala   |
| Dileep  |
| Sarjano |
| Leela   |
| Amal    |
+---------+
```

4.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 |
+---------+--------+-------------+
| Amala   |   5000 | Clerk       |
| Dileep  |  10000 | Manager     |
| Sarjano |   8000 | Sales man   |
| Leela   |   6000 | Accounting  |
| Amal    |   7000 | Finance     |
+---------+--------+-------------+
```

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

SELECT Ename,Designation FROM department WHERE Designation <>'Clerk' AND Designation <>'Manager';

```
+---------+-------------+
| Ename   | Designation |
+---------+-------------+
| Sarjano | Sales man   |
| Leela   | Accounting  |
| Amal    | Finance     |
+---------+-------------+
```

6.Display the name of all employees where the third letter of their name is an 'A'.

SELECT Ename FROM department WHERE Ename like '__a%';

```
+-------+
| Ename |
+-------+
| Amala |
| Amal  |
+-------+
```

SQL:6

Create a table customer with fields cid,cname,date_of_birth and place.

Create table loan with fields loan_no,cid and bname assigning suitable constraints.

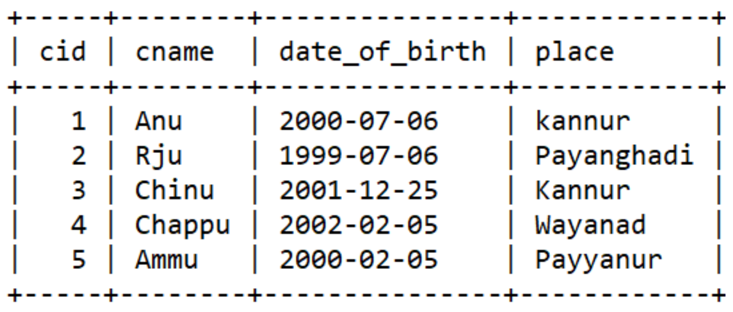
Create table depositor with fields accno,cid,balance and bname assigning suitable constraints.

Insert 5 records into each table.

```
CREATE TABLE customer(cid int primary key,cname varchar(20),date_of_birth date,place varchar(20));
```

```
INSERT INTO customer VALUES(1,'Anu','2000_07_06','kannur');

INSERT INTO customer VALUES(2,'Rju','1999_07_06','Payanghadi');

INSERT INTO customer VALUES(3,'Chinu','2001_12_25','Kannur');

INSERT INTO customer VALUES(4,'Chappu','2002_02_05','Wayanad');

INSERT INTO customer VALUES(5,'Ammu','2000_02_05','Payyanur');
```

```
SELECT * FROM customer;
```



```
+-----+--------+---------------+------------+
| cid | cname  | date_of_birth | place      |
+-----+--------+---------------+------------+
|   1 | Anu    | 2000-07-06    | kannur     |
|   2 | Rju    | 1999-07-06    | Payanghadi |
|   3 | Chinu  | 2001-12-25    | Kannur     |
|   4 | Chappu | 2002-02-05    | Wayanad    |
|   5 | Ammu   | 2000-02-05    | Payyanur   |
+-----+--------+---------------+------------+
```

```
CREATE TABLE loan(loan_no int primary key,cid int,foreign key(cid) references customer(cid),bname varchar(25));
```

```
INSERT INTO loan VALUES(2,0,'Panjab national bank');

INSERT INTO loan VALUES(4,2,'State bank of india');

INSERT INTO loan VALUES(8,0,'Central bank of india');

INSERT INTO loan VALUES(10,4,'Canara bank');
```

INSERT INTO loan VALUES(12,0,'Canara bank');

SELECT * FROM loan;

```
+---------+------+----------------------+
| loan_no | cid  | bname                |
+---------+------+----------------------+
|       2 |    1 | Panjab national bank |
|       4 |    2 | State bank of india  |
|       8 |    3 | Central bank of india |
|      10 |    4 | Canara bank          |
|      12 |    5 | Canara bank          |
+---------+------+----------------------+
```

CREATE TABLE depositor(cid int,foreign key(cid) references customer(cid),accno int primary key,balance int,bname varchar(25));

INSERT INTO depositor VALUES(1,101,0,'Panjab national bank');

INSERT INTO depositor VALUES(2,102,300,'State bank of india');

INSERT INTO depositor VALUES(3,103,3000,'Central bank of india');

INSERT INTO depositor VALUES(4,104,0,'Canara bank');

INSERT INTO depositor VALUES(5,105,0,'Canara bank');

SELECT * FROM depositor;

```
+------+-------+---------+----------------------+
| cid  | accno | balance | bname                |
+------+-------+---------+----------------------+
|    1 |   101 |    2000 | Panjab national bank |
|    2 |   102 |     300 | State bank of india  |
|    3 |   103 |    3000 | Central bank of india |
|    4 |   104 |    3000 | Canara bank          |
|    5 |   105 |     600 | Canara bank          |
+------+-------+---------+----------------------+
```

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

ALTER TABLE loan ADD amount int;

UPDATE loan SET amount=20000 WHERE cid=1;

UPDATE loan SET amount=30000 WHERE cid=2;

UPDATE loan SET amount=40000 WHERE cid=3;

UPDATE loan SET amount=50000 WHERE cid=4;

UPDATE loan SET amount=60000 WHERE cid=5;

SELECT * FROM loan;

```
+---------+------+----------------------+
| loan_no | cid  | bname                |
+---------+------+----------------------+
|       2 |    1 | Panjab national bank |
|       4 |    2 | State bank of india  |
|       8 |    3 | Central bank of india |
|      10 |    4 | Canara bank          |
|      12 |    5 | Canara bank          |
+---------+------+----------------------+
```

SELECT cname FROM customer WHERE cid=2;

```
+-------+
| cname |
+-------+
| Rju   |
+-------+
```

2.Calculator Rs 150 extra for all customers having loan. The added loan amount will display in a new column.

ALTER TABLE loan ADD updated_amount int;

UPDATE loan SET updated_amount=amount+150;

SELECT * FROM loan;

```
+---------+------+----------------------+--------+----------------+
| loan_no | cid  | bname                | amount | updated_amount |
+---------+------+----------------------+--------+----------------+
|       2 |    1 | Panjab national bank |  20000 |          20150 |
|       4 |    2 | State bank of india  |  30000 |          30150 |
|       8 |    3 | Central bank of india |  40000 |          40150 |
|      10 |    4 | Canara bank          |  50000 |          50150 |
|      12 |    5 | Canara bank          |  60000 |          60150 |
+---------+------+----------------------+--------+----------------+
```

3.Display loan_no,cname and place of customer who is residing in Kannur city.

SELECT loan_no,cname,place FROM customer INNER JOIN loan ON customer.cid=loan.cid AND Place='Kannur';

```
+---------+-------+--------+
| loan_no | cname | place  |
+---------+-------+--------+
|       2 | Anu   | kannur |
|       8 | Chinu | Kannur |
+---------+-------+--------+
```

4.Display all information from loan table for loan_no 2,8,10.

SELECT * FROM loan WHERE loan_no IN(2,8,10);

```
+---------+------+----------------------+
| loan_no | cid  | bname                |
+---------+------+----------------------+
|       2 |    1 | Panjab national bank |
|       8 |    3 | Central bank of india |
|      10 |    4 | Canara bank          |
+---------+------+----------------------+
```

5.Display all customers who have both loan and deposit.

DELETE FROM depositor WHERE cid=1;

DELETE FROM depositor WHERE cid=5;

DELETE FROM loan WHERE cid=1;

DELETE FROM loan WHERE cid=2;

DELETE FROM loan WHERE cid=5;

SELECT cname FROM customer JOIN loan ON customer.cid=loan.cid JOIN depositor ON customer.cid=depositor.cid;

```
+--------+
| cname  |
+--------+
| Chinu  |
| Chappu |
+--------+
```