DATA STRUCTURE LAB MANUAL

SECOND YEAR BCA

SEM-III

Prepared by

Prof. Deepa M. Mishra
Department of Information Technology
Sahyog College, Thane (W)

LIST OF PROGRAMS

PRACTICAL 1:

AIM: Program to insert node in the beginning and delete first node:

Solution:

```
#include<iostream>
using namespace std;
struct node
{
int data;
node *link;
};
node *start=NULL;
void insert()
{
node *n=new node;
cout<<"Enter data:";
cin>>n->data;
n->link=NULL;
if(start==NULL)
{
start=n;
cout<<"\n new node inserted \n";
}
else
{
node *r;
r=start;
while(r->link!=NULL)
{
r=r->link;
}
r->link=n;
cout<<"\n node inserted \n";
}
```

```
}
void del()
{
if(start==NULL)
{
cout<<"\n no node available \n";
}
else
{
node *t;
t=start;
start=start->link;
delete t;
cout<<"\n node deleted\n";
}
}
void display()
{
if(start==NULL)
{
cout<<"\n no node available \n";
}
else
{ node *r;
r=start;
while(r!=NULL)
{
cout<<r->data<<"->";
r=r->link;
}
cout<<"NULL\n";
}
}
int main()
```

```
{
int ch;
while(1)
{
cout<<"\n1.insert\n2.delete\n3.display\n";
cin>>ch;
if(ch==1)
insert();
else if(ch==2)
del();
else if(ch==3)
display();
else
exit(0);
}
}
```

Output:

```
1.insert
2.delete
3.display
1
Enter data:11

 new node inserted

1.insert
2.delete
3.display
1
Enter data:12

 node inserted

1.insert
2.delete
3.display
3
11->12->NULL

1.insert
2.delete
3.display
```

PRACTICAL 2:

AIM: Program to insert and delete node at specified position

Solution:

```
#include<iostream>
using namespace std;
struct node
{
int data;
node *link;
};
node *start=NULL;
int p=0,count=0;
void insert()
{
node *n=new node;
cout<<"\nEnter data:\n";
cin>>n->data;
n->link=NULL;
cout<<"enter position:";
cin>>p;//1
if(p==1 && start==NULL) //if list is empty
{
start=n;
count++;
cout<<"\n node inserted\n";
}
else if(p==1 && start!=NULL) //to insert data at 1st position if we have more
nodes available
{ node *u;
u=start;
start=n;
n->link=u;
count++;
```

```
cout<<"\n node inserted\n";
}
else if(p>count || p<=0)
{
cout<<"\n can not insert node\n";
}
else //to insert node at any position except 1st position
{
node *r,*t;
int i=1;
r=start;
while(i<p-1)
{
r=r->link;
i++;
}
t=r->link;
r->link=n;
n->link=t;
count++;
cout<<"\n node inserted\n";
}
}
//=========================================
void del()
{
cout<<"\nEnter position\n";
cin>>p;
if(p>count || p<=0 || start==NULL)
{
cout<<"\n no node available at this position\n";
}
else if(p==1) //to delete first node
{
```

```
node *t;
t=start;
start=t->link;
delete t;
count--;
cout<<"\n node deleted \n";
}

else
{
int i;
node *r,*t,*u;
r=start;
while(i<p-1)
{
r=r->link;
i++;
}
t=r->link;
u=t->link;
r->link=u;
delete t;
count--;
cout<<"\n node deleted \n";
cout<<"total node="<<count;
}
}
//=================================
void search()
{
int item;
node *r;
if(start==NULL)
{
```

```
cout<<"\n**no node available**\n";
}
else
{
cout<<"Enter item to search\n";
cin>>item;
r=start;
while(r!=NULL)
{
if(item==r->data)
{
cout<< item<<" found in the list\n";
break;
}
else
r=r->link;
}
if(r==NULL)
cout<<item<< " Not found in the list\n";
}
}
//==========================================
void display()
{
if(start==NULL)
{
cout<<"\n no node available \n";
}
else
{ node *r;
r=start;
while(r!=NULL)
{
cout<<r->data<<"->";
```

```
r=r->link;
}
cout<<"NULL\n";
}
cout<<"\ntotal node="<<count;
}
int main()
{
int ch;
while(1)
{
cout<<"\n1.insert\n2.delete\n3.search\n4.display\n";
cin>>ch;
switch(ch)
{
        case 1: insert();break;
        case 2: del();break;
        case 3: search();break;
        case 4: display();break;
        default: exit(0);}}}
```

Output:

```
1.insert
2.delete
3.search
4.display
1

Enter data:
11
enter position:1

 node inserted

1.insert
2.delete
3.search
4.display
1

Enter data:
12
enter position:2

 can not insert node

1.insert
2.delete
3.search
4.display
1

Enter data:
13
enter position:1

 node inserted

1.insert
2.delete
3.search
4.display
4
13->11->NULL
```

```
13->11->NULL

total node=2
1.insert
2.delete
3.search
4.display
1

Enter data:
23
enter position:2

 node inserted

1.insert
2.delete
3.search
4.display
4
13->23->11->NULL

total node=3
1.insert
2.delete
3.search
4.display
2

Enter position
3

 node deleted
total node=2
1.insert
2.delete
3.search
4.display
4
13->11->NULL
```

PRACTICAL 3:

AIM: Program to insert node at the end and delete last node

Solution:

```
#include<iostream>
using namespace std;
struct node
{
int data;
node *link;
};
node *start=NULL;
void insert()
{
node *n=new node;
cout<<"Enter data:";
cin>>n->data;
n->link=NULL;
if(start==NULL)
{
start=n;
cout<<"\n new node inserted \n";
}
else
{
node *r;
r=start;
while(r->link!=NULL)
{
r=r->link;
}
r->link=n;
cout<<"\n node inserted \n";
}
```

```
}
void del()
{
if(start==NULL)
{
cout<<"\n no node available \n";
}
else
{
node *t,*r;
if(start->link==NULL) //if only one node exists
{
delete start;
start=NULL;
cout<<"\n node deleted \n";
}
else
{
t=start;
while(t->link!=NULL) //to reach till last node
{
t=t->link;
}
r=start;
while(r->link!=t)
{
r=r->link;
}
r->link=NULL;
delete t;
cout<<"\n node deleted \n";
}
}
}
```

```
void display()
{
if(start==NULL)
{
cout<<"\n no node available \n";
}
else
{ node *r;
r=start;
while(r!=NULL)
{
cout<<r->data<<"->";
r=r->link;
}
cout<<"NULL\n";
}
}
int main()
{
int ch;
while(1)
{
cout<<"\n1.insert\n2.delete\n3.display\n";
cin>>ch;
if(ch==1)
insert();
else if(ch==2)
del();
else if(ch==3)
display();
else
exit(0);}}
```

Output:

```
1.insert
2.delete
3.display
1
Enter data:11

 new node inserted

1.insert
2.delete
3.display
1
Enter data:12

 node inserted

1.insert
2.delete
3.display
3
11->12->NULL
```

```
1.insert
2.delete
3.display
2

 node deleted

1.insert
2.delete
3.display
3
11->NULL

1.insert
2.delete
3.display
```

PRACTICAL 4:

AIM: Program for doubly linked list to insert and delete node at specified position.

Solution:

```
#include<iostream>
using namespace std;
struct node
{
int data;
node *next,*prev;
};
node *start=NULL;
int p=0,count=0;
void insert()
{
node *n=new node;
cout<<"\nEnter data:\n";
cin>>n->data;
n->next=NULL;
n->prev=NULL;
cout<<"enter position:";
cin>>p;//1
if(start==NULL) //if list is empty
{
start=n;
count++;
cout<<"\n node inserted\n";
}
else if(p==1 && start!=NULL) //to insert data at 1st position if we have more
nodes available
{
n->next=start;
start=n;
count++;
```

```
cout<<"\n node inserted\n";
}
else if(p>count || p<=0)
{
cout<<"\n can not insert node\n";
}
else //to insert node at any position except 1st position
{
node *r,*t,*q;
int i=1;
r=start;
while(i<p-1)
{
r=r->next;
i++;
}
q=r->next;
n->prev=r;
n->next=r->next;
r->next=n;
q->prev=n;

count++;
cout<<"\n node inserted\n";
}
}
//=========================================
void del()
{
cout<<"\nEnter position\n";
cin>>p;
if(p>count || p<=0 || start==NULL)
{
cout<<"\n no node available at this position\n";
```

```
}
else if(p==1) //to delete first node
{
node *t;
t=start;
start=t->next;
delete t;
count--;
cout<<"\n node deleted \n";
}

else
{
int i;
node *r,*t,*u;
r=start;
while(i<p-1)
{
r=r->next;
i++;
}
t=r->next;
r->next=t->next;
delete t;
count--;
cout<<"\n node deleted \n";
cout<<"total node="<<count;
}
}
//=====================================
void search()
{
int item;
node *r;
```

```
if(start==NULL)
{
cout<<"\n**no node available**\n";
}
else
{
cout<<"Enter item to search\n";
cin>>item;
r=start;
while(r!=NULL)
{
if(item==r->data)
{
cout<< item<<" found in the list\n";
break;
}
else
r=r->next;
}
if(r==NULL)
cout<<item<< " Not found in the list\n";
}
}
//=========================================
void display()
{
if(start==NULL)
{
cout<<"\n no node available \n";
}
else
{ node *r;
r=start;
while(r!=NULL)
```

```
{
cout<<"|"<<r->data<<"| ";
r=r->next;
}
cout<<"NULL\n";
}
cout<<"\ntotal node="<<count;

}
int main()
{
int ch;
while(1)
{
cout<<"\n1.insert\n2.delete\n3.search\n4.display\n";
cin>>ch;
switch(ch)
{
        case 1: insert();break;
        case 2: del();break;
        case 3: search();break;
        case 4: display();break;
        default: exit(0);
}
}
}
```

Output:

```
1.insert
2.delete
3.search
4.display
1

Enter data:
11
enter position:1

 node inserted

1.insert
2.delete
3.search
4.display
1

Enter data:
43
enter position:1

 node inserted

1.insert
2.delete
3.search
4.display
1

Enter data:
43
enter position:2

 node inserted

1.insert
2.delete
3.search
4.display
4
|43| |43| |11| NULL
```

```
|43| |43| |11| NULL

total node=3
1.insert
2.delete
3.search
4.display
2

Enter position
2

 node deleted
total node=2
1.insert
2.delete
3.search
4.display
4
|43| |11| NULL

total node=2
1.insert
2.delete
3.search
4.display
```

PRACTICAL 5:

AIM: Program to insert node in the beginning, end and delete last node from doubly linked list

Solution:

```
#include<iostream>
using namespace std;
struct node
{ node *prev;
int data;
node *next;
};

node *start=NULL;
//=========================
void insert_beg()
{
node *n=new node;
cout<<"\nEnter data:\n";
cin>>n->data;
n->prev=NULL;
n->next=NULL;
if(start==NULL)
{
start=n;
cout<<"\n node inserted \n";
}
else
{
node *r;
r=start;
start=n;
n->next=r;
cout<<"\n node inserted \n";
```

```
}
}
//=======================
void insert_end()
{
node *n=new node;
cout<<"\nEnter data:\n";
cin>>n->data;
n->prev=NULL;
n->next=NULL;
if(start==NULL)
{
start=n;
cout<<"\n node inserted \n";
}
else
{
node *r;
r=start;
while(r->next!=NULL)
{
r=r->next;
}
r->next=n;
cout<<"\n node inserted \n";
}
}
//=======================
void display()
{
if(start==NULL)
{
cout<<"\n no node available \n";
}
```

```
else
{ node *r;
r=start;
while(r!=NULL)
{
cout<<r->data<<"->";
r=r->next;
}
cout<<"NULL\n";
}
}
//=========================
void del_beg()
{
if(start==NULL)
{
cout<<"\n can not delete\n";
}
else if(start->next==NULL) //if only one node exists
{
node *p;
p=start;
start=NULL;
delete p;
cout<<"\n node deleted \n";
}
else //if more than one node exists
{
node *t;
t=start;
start=start->next;
start->prev=NULL;
delete t;
cout<<"\n node deleted\n";
```

```
}
}
//==========================
int main()
{
int ch;
while(1)
{
cout<<"\n1.insert begining\n2.insert end\n3.delete begining\n4.display\n";
cin>>ch;
if(ch==1)
insert_beg();
else if(ch==2)
insert_end();
else if(ch==3)
del_beg();
else if(ch==4)
display();
else
exit(0);
}
}
```

Output:

```
1.insert begining
2.insert end
3.delete begining
4.display
1

Enter data:
12

 node inserted

1.insert begining
2.insert end
3.delete begining
4.display
2

Enter data:
43

 node inserted

1.insert begining
2.insert end
3.delete begining
4.display
1

Enter data:
54

 node inserted
```

```
1.insert begining
2.insert end
3.delete begining
4.display
4
54->12->43->NULL

1.insert begining
2.insert end
3.delete begining
4.display
3

 node deleted

1.insert begining
2.insert end
3.delete begining
4.display
4
12->43->NULL

1.insert begining
2.insert end
3.delete begining
4.display
```

PRACTICAL 6:

AIM: Program to implement circular linked list

Solution:

```
#include<iostream>
using namespace std;
struct node
{
int data;
node *link;
};
node *start=NULL;
int p=0,c=0;
//=========
void insert_beg()
{
node *n=new node;
cout<<"enter data\n";
cin>>n->data;
n->link=NULL;
if(start==NULL)
{
start=n;
n->link=start;
cout<<"Node inserted\n";
c++;
}
else
{
node *r,*t;
r=t=start;
start=n;
n->link=t;
while(r->link!=t)
```

```
{
r=r->link;
}
r->link=start;
cout<<"Node inserted\n";
c++;
}
}
//===========
void insert_end()
{
node *n=new node;
cout<<"enter data\n";
cin>>n->data;
n->link=NULL;
if(start==NULL)
{
start=n;
n->link=start;
cout<<"Node inserted\n";
c++;
}
else
{
node *r;
r=start;
while(r->link!=start)
{
r=r->link;
}
r->link=n;
n->link=start;
cout<<"Node inserted\n";
c++;
```

```
}
}
//===========
void insert_mid()
{
node *n=new node;
cout<<"enter data\n";
cin>>n->data;
n->link=NULL;
cout<<"\nEnter position\n";
cin>>p;
if(p==1 && start==NULL)
{
start=n;
n->link=start;
c++;
cout<<"\n node insreted \n";
}
else if(p==1 && start!=NULL)
{
node *r;
r=start;
while(r->link!=start)
{
r=r->link;
}
r->link=n;
n->link=start;
start=n;
c++;
cout<<"\n node insreted \n";
}
else if(p>c || p<=0)
{
```

```
cout<<"\n can not delete\n";
}
else
{
int i=1;
node *r=start;
while(i<p-1)
{
r=r->link;
i++;
}
n->link=r->link;
r->link=n;
c++;
cout<<"\n node inserted \n";
}
}
//===========
void del_beg()
{
if(start==NULL)
{
cout<<"node not available\n";
}
else if(start->link==start)
{
delete start;
start=NULL;
cout<<"\n node deleted \n";
c--;
}
else
{
node *r,*q;
```

```
q=r=start;
start=start->link;
while(r->link!=q)
{
r=r->link;
}
r->link=start;
delete q;
cout<<"\n node deleted \n";
c--;
}
}
void del_end()
{
if(start==NULL)
{
cout<<"node not available\n";
}
else if(start->link==start)
{
delete start;
start=NULL;
cout<<"\n node deleted \n";
c--;

}
else
{
node *r,*t;
r=start;
while(r->link!=start)
{
r=r->link;
}
```

```
t=start;
while(t->link!=r)
{
t=t->link;
}
t->link=start;
delete r;
cout<<"\n node deleted \n";
c--;
}
}
//==========
void del_mid()
{
int i=1;
cout<<"\n Enter position:\n";
cin>>p;
if(p>c || start==NULL || p<1)
{

cout<<"node not available\n";

}
else if(p==1 && start->link==start)
{

delete start;

start=NULL;
cout<<"\n node deleted \n";
c--;
}
else if(p==1 && start!=NULL)
{
```

```
node *r,*q;
q=r=start;
start=start->link;
while(r->link!=q)
{
r=r->link;
}
r->link=start;
delete q;
cout<<"\n node deleted \n";
c--;
}
else
{
node *r,*q;
r=start;
while(i<p-1)
{
r=r->link;
i++;
}
q=r->link;
r->link=q->link;
c--;
cout<<"\n node deleted \n";
}
}
//==========
void display()
{
if(start==NULL)
{
cout<<"node not available\n";
}
```

```
else
{
node *r;
r=start;
do
{
cout<<r->data<<" ";
r=r->link;
}while(r!=start);
}
}
//=========
int main()
{
int ch;
while(1)
{
cout<<"\n1.insert beginning\n2.insert end\n3.delete beginning\n4.delete
end\n5.dispaly\n6.insert_mid\n7.delete mid\n";
cin>>ch;
switch(ch)
{
 case 1: insert_beg(); break;
 case 2: insert_end(); break;
 case 3: del_beg();   break;
 case 4: del_end();  break;
 case 5: display(); break;
 case 6: insert_mid();break;
 case 7: del_mid();break;
 default: exit(0);
}
```

PRACTICAL 7:

AIM: Program to implement stack using array

Solution:

```
#include<iostream>
using namespace std;
#define size 5
int s[size],top=-1;
void push()
{
if(top==size-1)
{
cout<<"stack full can not insert element\n";
}
else
{
top++;
cout<<"Enter element to be pushed\n";
cin>>s[top];
cout<<"Element inserted\n";
}
}
//==============================================
void pop()
{
if(top==-1)
{
cout<<"stack empty can not delete\n";
}
else
{
top--;
cout<<"Element removed\n";
}
```

```
}
//=====================================================
void display()
{
int i;
if(top==-1)
{
cout<<"stack empty \n";
}
for(i=top;i>=0;i--)
{
cout<<s[i]<<"\n";
}
}
int main()
{
int ch;
while(1)
{
cout<<"\n1. for push()\n2.for pop()\n3.for display()\n4.for exit()\n";
cin>>ch;
if(ch==1)
push();
else if(ch==2)
pop();
else if(ch==3)
display();
else
exit(0);
}
return 0;
}
```

Output:

```
1.for push()
2.for pop()
3.for display()
4.for exit()
1
Enter element to be pushed
14
Element inserted

1.for push()
2.for pop()
3.for display()
4.for exit()
1
Enter element to be pushed
16
Element inserted

1.for push()
2.for pop()
3.for display()
4.for exit()
3
16
14
```

```
1.for push()
2.for pop()
3.for display()
4.for exit()
2
Element removed

1.for push()
2.for pop()
3.for display()
4.for exit()
3
14

1.for push()
2.for pop()
3.for display()
4.for exit()
2
Element removed

1.for push()
2.for pop()
3.for display()
4.for exit()
3
stack empty
```

PRACTICAL 8:

AIM: Program to implement stack using linked list

Solution:

```
#include<iostream>
using namespace std;
struct node
{
        int info;
        struct node *link;
};
node *top=NULL;
void push()
{
        node *t=new node;
        node *temp;
        printf("Enter the element to be pushed\n");
        cin>>t->info;
        t->link=NULL;
        if(top==NULL)
        {
          top=t;
          top->link=NULL;
        }
        else
        {
           temp=top;
           top=t;
           top->link=temp;
        }
        cout<<"Item Inserted\n";
}
void pop()
{       node *d;
```

```
        if(top==NULL)
        {
                printf("stack empty");
        }
        else
        {
                d=top;
                top=d->link;
                delete d;
        cout<<"Item popped out\n";
        }
}
void display()
{
        node *r;
        if(top==NULL)
        {
                cout<<"stack empty";
        }
        else
        {
          r=top;
          while(r!=NULL)  //inserting node at front
          {
                cout<<"| " <<r->info<< " |\n";
                cout<<"|___|\n";
                r=r->link;
        }
        }
}

int main()
{
 int ch;
```

```
 while(1)
 {
 cout<<"\n1. for push()\n2.for pop()\n3.for display()\n4.for exit()\n";
 cin>>ch;
 if(ch==1)
 push();
 else if(ch==2)
 pop();
 else if(ch==3)
 display();
 else
 exit(0);
}
 return 0;
}
```

Output:

```
1. for push()
2.for pop()
3.for display()
4.for exit()
1
Enter the element to be pushed
18
Item Inserted

1. for push()
2.for pop()
3.for display()
4.for exit()
1
Enter the element to be pushed
19
Item Inserted

1. for push()
2.for pop()
3.for display()
4.for exit()
3
| 19 |
|___|
| 18 |
|___|
| 15 |
|___|
```

```
1. for push()
2.for pop()
3.for display()
4.for exit()
2
Item popped out

1. for push()
2.for pop()
3.for display()
4.for exit()
3
| 18 |
|___|
| 15 |
|___|

1. for push()
2.for pop()
3.for display()
4.for exit()
```

PRACTICAL 9:

AIM: Program to implement linear queue using array

Solution:

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

int f=-1,r=-1;
#define size 6
int q[size];

//==============
void enqueue()
{
int data;
cout<<"enter data\n";
cin>>data;
if(f==-1)
{
        f=r=0;
        q[r]=data;
        cout<<"\ndata inserted\n";
}
else if(r==size-1)
{
        cout<<"\nqueue full\n";
}
else
{
        r++;
        q[r]=data;
        cout<<"\ndata inserted\n";
}
}
```

```
//================
void dequeue()
{
        if(f==-1 && r==-1)
        {
                cout<<"\n queue empty";
        }
        else if(f==r)
        {
                f=r=-1;
                cout<<"\n data deleted\n";
        }
        else
        {
                f++;
                cout<<"\n data deleted\n";
        }
}
//==============
void display()
{
        if(f==-1 && r==-1)
        {
                cout<<"\n queue empty";
        }
        else
        {
        for(int i=f;i<=r;i++)
        {
                cout<<q[i]<<" ";
        }
        }

}
```

```
//==============
int main()
{
        int ch;
        while(1)
        {
                cout<<"\n1.enqueue 2.dequeue 3.display 4.exit\n";
                cin>>ch;
                switch(ch)
                {
                        case 1: enqueue();break;
                        case 2: dequeue();break;
                        case 3: display();break;
                        default: exit(0);
                }
        }
}
```

Output:

```
1.enqueue 2.dequeue 3.display 4.exit
1
enter data
12

data inserted

1.enqueue 2.dequeue 3.display 4.exit
1
enter data
13

data inserted

1.enqueue 2.dequeue 3.display 4.exit
1
enter data
14

data inserted

1.enqueue 2.dequeue 3.display 4.exit
3
12 13 14
1.enqueue 2.dequeue 3.display 4.exit
```

```
1.enqueue 2.dequeue 3.display 4.exit
2

 data deleted

1.enqueue 2.dequeue 3.display 4.exit
2

 data deleted

1.enqueue 2.dequeue 3.display 4.exit
3
14
1.enqueue 2.dequeue 3.display 4.exit
```

PRACTICAL 10:

AIM: Program to implement circular queue using array

Solution:

```
#include<stdio.h>
#include<stdlib.h>
#define size 5

int q[size],front=-1,rear=-1;
void enqueue()
{
if((rear==size-1 && front==0) || rear==front-1)//full
{
printf("\n queue is full \n");
}
else if(front==-1 && rear==-1)//1st data
{
front=0;
rear=0;
printf("\n enter data \n");
scanf("%d",&q[rear]);
printf("\n data inserted \n ");
}
else if(rear==size-1 && front!=0)//circular
{
rear=0;
printf("\n enter data \n");
scanf("%d",&q[rear]);
printf("\n data inserted \n ");
}
else
{
rear++;
```

```
printf("\n enter data \n");
scanf("%d",&q[rear]);
printf("\n data inserted \n ");
}
}
void dequeue()
{
if(front==-1 && rear==-1)//empty
{
printf("\n queue is empty\n");
}
else if(front==rear)//last element
{
front=-1;
rear=-1;
printf("\n data deleted \n");
}
else if (front !=size-1)
{
front++;
printf("\n data deleted \n");
}
else if(front==size-1 && rear>=0)//circular
{
front=0;
printf("\n data deleted \n");
}
}
void display()
{
int i,j;
if(front==-1 && rear==-1)
printf("\n queue is empty \n");
if(rear<front)
```

```
{
for(i=front;i<=size-1;i++)
{
printf("%d ",q[i]);
}
for(j=0;j<=rear;j++)
{
printf("%d ",q[j]);
}
}
else
{
for(i=front;i<=rear;i++)
{
printf("%d ",q[i]);
}}}
int main()
{
int ch;
while(1)
{
printf("\n1.insert\n2.delete\n3.dispaly\n4.exit\n");
scanf("%d",&ch);
if(ch==1)
enqueue();
else if(ch==2)
dequeue();
else if(ch==3)
display();
else
exit(0);
}
}
```

PRACTICAL 11:

AIM: Program to implement priority queue using linked list

Solution:

```
#include<iostream>
using namespace std;
struct node
{
int data;
int pri;
node *link;
};
node *start=NULL;
void insert()
{
        node *n=new node;
        cout<<"Enter data:";
        cin>>n->data;
        cout<<"Enter priority:";
        cin>>n->pri;
        n->link=NULL;
        if(start==NULL)
        {
                start=n;
                cout<<"\n new node inserted \n";
        }
        else if(start->pri > n->pri) //to insert in begining
        {
                n->link=start;
                start=n;
                cout<<"\n new node inserted \n";
        }
        else
        {
```

```
                node *r=start,*temp;
                while(r!=NULL && r->pri <= n->pri)
                {
                        temp=r;
                        r=r->link;
                }
                temp->link=n;
                n->link=r;
                cout<<"\n new node inserted \n";
        }
}
void del()
{
        if(start==NULL)
        {
                cout<<"\n no node available \n";
        }
        else
        {
           node *t;
          t=start;
          start=start->link;
          delete t;
          cout<<"\n node deleted\n";
        }
}
void display()
{
        if(start==NULL)
        {
                cout<<"\n no node available \n";
        }
        else
        {       node *r;
```

```
                r=start;
                while(r!=NULL)
                {
                        cout<<"|"<<r->data<<"->"<<r->pri<<"|";
                        r=r->link;
                }
                cout<<"NULL\n";
                }
}
int main()
{
        int ch;
        while(1)
        {
        cout<<"\n1.insert\n2.delete\n3.display\n4.Exit\n";
        cin>>ch;
        if(ch==1)
        insert();
        else if(ch==2)
        del();
        else if(ch==3)
        display();
        else
        exit(0);
        }
}
```

Output:

```
1.insert
2.delete
3.display
4.Exit
1
Enter data:11
Enter priority:2

 new node inserted

1.insert
2.delete
3.display
4.Exit
1
Enter data:12
Enter priority:4

 new node inserted

1.insert
2.delete
3.display
4.Exit
3
|11->2||12->4|NULL
```

```
 new node inserted

1.insert
2.delete
3.display
4.Exit
3
|32->1||11->2||12->4||44->6|NULL

1.insert
2.delete
3.display
4.Exit
2

 node deleted

1.insert
2.delete
3.display
4.Exit
3
|11->2||12->4||44->6|NULL
```

PRACTICAL 12:

AIM: Program to implement doubly ended queue

Solution:

```
#include<iostream>
using namespace std;
#define Size 5

int deque_arr[Size];
int front = -1;
int rear = -1;

/*Begin of insert_rear*/
void insert_rear()
{
   int added_item;
   if((front == 0 && rear == Size-1) || (front == rear+1))
   {   cout<<"Queue Overflow\n";
       return;}
   if (front == -1)  /* if queue is initially empty */
   {   front = 0;
       rear = 0;}
   else
   if(rear == Size-1)  /*rear is at last position of queue */
       rear = 0;
   else
       rear = rear+1;

   cout<<"Input the element for adding in queue : ";
   cin>>added_item;
   deque_arr[rear] = added_item ;
}
/*End of insert_rear*/
```

```
/*Begin of insert_front*/
void insert_front()
{   int added_item;
    if((front == 0 && rear == Size-1) || (front == rear+1))
    { cout<<"Queue Overflow \n";
        return;  }
    if (front == -1)/*If queue is initially empty*/
    {   front = 0;
        rear = 0;    }
    else
    if(front== 0)
        front=Size-1;
    else
        front=front-1;
    cout<<"Input the element for adding in queue : ";
    cin>>added_item;
    deque_arr[front] = added_item ;  }
/*End of insert_front*/

/*Begin of delete_front*/
void delete_front()
{   if (front == -1)
    {   cout<<"Queue Underflow\n";
        return ;
    }
   cout<<"Element deleted from queue is :"<<deque_arr[front]<<endl;
    if(front == rear) /*Queue has only one element */
    {   front = -1;
        rear=-1;
    }
    else
        if(front == Size-1)
            front = 0;
        else
```

```
        front = front+1;
}
/*End of delete_front*/

/*Begin of delete_rear*/
void delete_rear()
{
   if (front == -1)
   {
      cout<<"Queue Underflow\n";
      return ;
   }
   cout<<"Element deleted from queue is :"<<deque_arr[rear]<<endl;
   if(front == rear) /*queue has only one element*/
   {
      front = -1;
      rear=-1;
   }
   else
      if(rear == 0)
         rear=Size-1;
      else
         rear=rear-1;    }
/*End of delete_rear*/

/*Begin of input_que*/
void display_queue()
{
   int front_pos = front,rear_pos = rear;

   if(front == -1)
   {   cout<<"Queue is empty\n";
      return;
   }
```

```
    cout<<"Queue elements :\n";
    if( front_pos <= rear_pos )
    {
        while(front_pos <= rear_pos)
        {
            cout<<deque_arr[front_pos];
            front_pos++;
        }
    }
    else
    {
        while(front_pos <= Size-1)
        {   cout<<deque_arr[front_pos];
            front_pos++;
        }
        front_pos = 0;
        while(front_pos <= rear_pos)
        {
           cout<<deque_arr[front_pos];
            front_pos++;
        }
    }
    cout<<endl;
}

void input_que()
{   int choice;
    do
    {   cout<<"1.Insert at rear\n";
         cout<<"2.Delete from front\n";
         cout<<"3.Delete from rear\n";
        cout<<"4.Display\n";
        cout<<"5.Quit\n";
         cout<<"Enter your choice : ";
```

```
      cin>>choice;

       switch(choice)
       {   case 1:
           insert_rear();
           break;
        case 2:
           delete_front();
           break;
        case 3:
           delete_rear();
           break;
        case 4:
           display_queue();
           break;
        case 5:
           break;
        default:
           printf("Wrong choice\n");
       }
   }while(choice!=5);
}

void output_que()
{   int choice;
    do
    {   cout<<"1.Insert at rear\n";
        cout<<"2.Delete from front\n";
        cout<<"3.Delete from rear\n";
       cout<<"4.Display\n";
       cout<<"5.Quit\n";
        cout<<"Enter your choice : ";
       cin>>choice;
       switch(choice)
```

```
        {
         case 1:
            insert_rear();
            break;
         case 2:
            insert_front();
            break;
         case 3:
            delete_front();
            break;
         case 4:
            display_queue();
            break;
         case 5:
            break;
         default:
           cout<<"Wrong choice\n";
        }
    }while(choice!=5);
}

main()
{
     int choice;
    cout<<"1.Input restricted dequeue\n";
    cout<<"2.Output restricted dequeue\n";
    cout<<"Enter your choice : ";
    cin>>choice;
    switch(choice)
    {
     case 1 :
        input_que();
        break;
     case 2:
```

```
        output_que();
        break;
     default:
        cout<<"Wrong choice\n";
    }
}
```

PRACTICAL 13:

AIM: Program to calculate factorial using recursion

Solution:

```
#include<iostream>
using namespace std;
int fact(int);
int main()
{
int a,f;
cout<<"Enter a number:";
cin>>a;
f=fact(a);
cout<<"factorial is "<<f;
return 0;
}
int fact(int n)
{
if(n==0)
return 1;
else
return n*fact(n-1);
}
```

Output:

```
Enter a number:6
factorial is 720
--------------------------------
```

PRACTICAL 14:

AIM: Program to solve tower of Hanoi Problem using recursion

Solution:

```
#include<iostream>
using namespace std;
void towers(int, char, char, char);
int main()
{
int num;
char src='A', aux='B', dest='C';
cout<<"Enter the number of disks : ";
cin>>num;
cout<<"The sequence of moves involved in the Tower of Hanoi are :\n";
towers (num, src,dest,aux);
return 0;
}
void towers( int num, char source, char dest, char auxpeg)
{
if (num == 1)
cout<<"\n Move disk 1 from peg "<<source<<" to peg "<< dest;
else
{
towers (num - 1, source, auxpeg, dest);
cout<<"\n Move disk "<< num<<" from peg " <<source<<" to peg "<< dest;
towers (num - 1, auxpeg, dest, source);
}
}
```

Output:

```
Enter the number of disks : 3
The sequence of moves involved in the Tower of Hanoi are :

 Move disk 1 from peg A to peg C
 Move disk 2 from peg A to peg B
 Move disk 1 from peg C to peg B
 Move disk 3 from peg A to peg C
 Move disk 1 from peg B to peg A
 Move disk 2 from peg B to peg C
 Move disk 1 from peg A to peg C
--------------------------------
```

PRACTICAL 15:

AIM: Program to search an element using linear Search

Solution:

```
#include<iostream>
using namespace std;
int main()
{
        int a[10],n,i, data,c=0;
        cout<<"enter no of elements:\n";
        cin>>n;
        cout<<"enter elements:\n";
        for(i=0;i<n;i++)
        {
                cin>>a[i];
        }
    cout<<"Enter element to search:";
        cin>>data;
        for(i=0;i<n;i++)
        {
                if(data==a[i])
                {
                        c++;
                        cout<<"Data found";
                        break;
                }
        }
        if(c==0)
        cout<<"data not found";
}
```

Output:

```
enter no of elements:
5
enter elements:
12
32
44
5
65
Enter element to search:
44
Data found
```

```
enter no of elements:
4
enter elements:
12
32
44
55
Enter element to search:
66
data not found
```

PRACTICAL 16:

AIM: Program to search an element using binary Search

Solution:

```
#include<iostream>
using namespace std;
int main()
{
        int n,search;
        cout<<"enter no of elements:";
        cin>>n;

        int a[n],l=0,r=n-1,m;

        cout<<"Enter elements in sorted order:";
        for(int i=0;i<n;i++)
        {
                cin>>a[i];
        }
        cout<<"enter data to search:";
        cin>>search;
        m=(l+r)/2;
        while(l<=r)
        {
                if(search==a[m])
                {
                        cout<<"data found:";break;
                }
                else if(search < a[m])
                {
                        r=m-1;
                }
                else
                {
                   l=m+1;
                }
                m=(l+r)/2;
```

```
        }
        if(l>r)
        {
                cout<<"data not found:";
        }
}
```

Output:

```
enter no of elements:5
Enter elements in sorted order:66
77
78
98
99
enter data to search:77
data found:
```

```
enter no of elements:4
Enter elements in sorted order:11
23
56
76
enter data to search:79
data not found:
```

PRACTICAL 17:

AIM: Program to sort array elements using bubble sort

Solution:

```
#include<stdio.h>
int main()
{
        int a[20],n, i, temp;
        printf("enter no of elements:");
        scanf("%d", &n);
        printf("Enter Elements:");
        for(int i=0;i<n;i++)
        scanf("%d",&a[i]);
        printf("sorted elements are\n");
        printf("Elements are\n");
        for(int i=0;i<n;i++)
        {
                printf("%d ",a[i]);
        }
}
```

Output:

```
enter no of elements:6
Enter Elements:
12
45
4
65
7
66
sorted elements are
Elements are
4 7 12 45 65 66
```

PRACTICAL 18:

AIM: Program to sort array elements using selection sort

Solution:

```
#include<stdio.h>
int main()
{
        int a[20],n, i, temp,min;
        printf("enter no of elements:");
        scanf("%d",&n);
        printf("Enter Elements:");
        for(int i=0;i<n;i++)
        scanf("%d",&a[i]);

        for(i=0;i<=n-1;i++)
        {
                min=i;
                for(int j=i+1;j<=n-1;j++)
                {
                        if(a[min]>a[j])
                        {
                         min=j; //min will hold index no smallest element
                        }
                }
                        temp=a[i];
                        a[i]=a[min];
                        a[min]=temp;
        }
        printf("Elements are\n");
        for(int i=0;i<n;i++)
        {
                printf("%d ",a[i]);
        }
}
```

Output:

```
enter no of elements:7
Enter Elements:23
4
28
7
33
5
233
Elements are
4 5 7 23 28 33 233
```

PRACTICAL 19:

AIM: Program to sort array elements using insertion sort

Solution:

```
#include<iostream>
using namespace std;
int main()
{
	int a[10],n;
	int temp,j;
	cout<<"enter no of elements:\n";
	cin>>n;
	cout<<"enter data\n";
	for(int i=0;i<n;i++)
	{
		cin>>a[i];
	}
	for(int i=1;i<n;i++)
	{
		temp=a[i];
		j=i-1;
		while(j>=0 && temp<=a[j])
		{
			a[j+1]=a[j];
			j--;
		}
		a[j+1]=temp;
	}
	cout<<"\nsorted data\n";
	for(int i=0;i<n;i++)
	{
	cout<<a[i]<<" ";
	}
}
```

Output:

```
enter no of elements:
6
enter data

12
3
43
56
7
32

sorted data
3 7 12 32 43 56
```

PRACTICAL 20:

AIM: Program to sort array elements using merge sort

Solution:

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

/* Function to merge the subarrays of a[] */
void merge(int a[], int beg, int mid, int end)
{
   int i, j, k;
   int n1 = mid - beg + 1;
   int n2 = end - mid;
   int LeftArray[n1], RightArray[n2]; //temporary arrays

   /* copy data to temp arrays */
   for (int i = 0; i < n1; i++)
   LeftArray[i] = a[beg + i];
   for (int j = 0; j < n2; j++)
   RightArray[j] = a[mid + 1 + j];

   i = 0; /* initial index of first sub-array */
   j = 0; /* initial index of second sub-array */
   k = beg;  /* initial index of merged sub-array */

   while (i < n1 && j < n2)
   {
      if(LeftArray[i] <= RightArray[j])
      {
         a[k] = LeftArray[i];
         i++;
      }
```

```
        else
        {
            a[k] = RightArray[j];
            j++;
        }
        k++;
    }
    while (i<n1)
    {
        a[k] = LeftArray[i];
        i++;
        k++;
    }
    while (j<n2)
    {
        a[k] = RightArray[j];
        j++;
        k++;
    }
}
void mergeSort(int a[], int beg, int end)
{
    if (beg < end)
    {
        int mid = (beg + end) / 2;
        mergeSort(a, beg, mid);
        mergeSort(a, mid + 1, end);
        merge(a, beg, mid, end);
    }
}
/* Function to print the array */
void printArray(int a[], int n)
```

```
{
    int i;
    for (i = 0; i < n; i++)
    cout<<a[i]<<" ";
}
int main()
{
    int a[20] ;
    int n ;
    cout<<"enter no of elements:";
    cin>>n;
    cout<<"enter elements:";
    for(int i=0;i<n;i++)
    {
        cin>>a[i];
     }
    cout<<"Before sorting array elements are - \n";
    printArray(a, n);
    mergeSort(a, 0, n - 1);
    cout<<"\nAfter sorting array elements are - \n";
    printArray(a, n);
    return 0;
}
```

Output:

```
enter no of elements:4
enter elements:43
5
66
7
Before sorting array elements are -
43 5 66 7
After sorting array elements are -
5 7 43 66
--------------------------------
```

PRACTICAL 21:

AIM: Program to sort array elements using quick sort

Solution:

```
#include<iostream>
using namespace std;
void Quick_Sort(int a[10],int lb,int ub)
{
        int start,end,pivot,temp;
        if(lb<ub)
        {
        start=lb;
        end=ub;
        pivot=lb;

        while(start<end)
        {
                while(a[start]<=a[pivot])
        {
                start++;
        }
        while(a[end]>a[pivot])
        {
                end--;
        }
        if(start<end)
        {
                temp=a[start];
                a[start]=a[end];
                a[end]=temp;
        }
        else
        {
                temp=a[pivot];
```

```
                a[pivot]=a[end];
                a[end]=temp;
                Quick_Sort(a,lb,end-1);
                Quick_Sort(a,end+1,ub);
        }
    }
}
}
int main()
{
        int a[20],n;
        cout<<"enter no of elements:";
        cin>>n;
        cout<<"Enter elements:";
        for(int i=0;i<n;i++)
        {
                cin>>a[i];
        }
        Quick_Sort(a,0,n-1);
        cout<<"Elements are\n:";
        for(int i=0;i<n;i++)
        {
                cout<<a[i]<<" ";
        }
}
```

Output:

```
enter no of elements:5
Enter elements:
54
34
12
32
5
Elements are
:5 12 32 34 54
--------------------------------
```