1.1Simple_Queue_13

Code:

```
#include<iostream>
using namespace std;
class nick
{
public:
int front,rear,choice,num,size,s[10],ele;
public:
void enqueue()
{
cout<<"enqueue operation is selected\n";
if(rear==size-1)
{
cout<<"queue exausted........\n";
}
else
{
cout<<"enter the element you want to push into the queue\n";
cin>>num;
++rear;
s[rear]=num;
cout<<"the number is inserted into the queue\n";
}
}

void dequeue()
{
cout<<"you have selected dequeue operation\n";
if(front>=rear)
{
```

```
cout<<"no element present in the stack\n";
}
else
{
ele=s[front];
front=front+1;
cout<<"you have removed "<<ele<<" front the queue....";
}
}

void dis()
{

cout<<"display option is selected\n";
if(front>rear)
{
cout<<"no element is present in array\n";
}
else
{
for(int i=front;i<=rear;i++)
{
cout<<i<<" : "<<s[i]<<"\n";
}
}
}

void get()
{
front=0;
rear=-1;
```

Queue programs

```
cout<<"enter the size of the queue:-\n";
cin>>size;
int s[size];
do
{
cout<<"enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-\n";
cin>>choice;
switch(choice)
{
case 1:
enqueue();
break;

case 2:
dequeue();
break;

case 3:
dis();
break;

case 4:
break;

default:
cout<<"invalid choice ## enter right choice.....\n";
}
}
while(choice!=4);
}
};
```

Queue programs

```
int main()
{
nick o;
o.get();
}
```

Output:--

```
enter the size of the queue:-
3
enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-
1
enqueue operation is selected
enter the element you want to push into the queue
1
the number is inserted into the queue
enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-
1
enqueue operation is selected
enter the element you want to push into the queue
2
the number is inserted into the queue
enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-
1
enqueue operation is selected
enter the element you want to push into the queue

3
the number is inserted into the queue
```

```
enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-

3
display option is selected
0 : 1
1 : 2
2 : 3
enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-

1
enqueue operation is selected
queue exausted........
enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-
2
you have selected dequeue operation
you have removed 1 front the queue....enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-
2
you have selected dequeue operation
you have removed 2 front the queue....enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-
2
you have selected dequeue operation
no element present in the stack
enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-
```

1.2Doubly_Ended_Queue_13

Queue programs

Code:

```
#include<iostream>
using namespace std;
class nick
{
public:
int front,rear,choice,num,size,s[10],ele;
public:
void enqueue()
{
cout<<"enqueue operation is selected\n";
int c1;
cout<<"from where u wanna insert the element:(1:front 2:rear):-";
cin>>c1;
switch(c1)
{
case 1:
{
cout<<"front enqueue\n";
if(front==-1)
{
front=0;
rear=0;
cout<<"enter the element you want to push into the queue\n";
cin>>num;
s[rear]=num;
cout<<"the number is inserted into the queue\n";
}
else
{
if(front==0)
```

```
{
front = size;
}
if((front-1)%size==rear)
{
cout<<"queue exausted........\n";
}
else
{
cout<<"enter the element you want to push into the queue\n";
cin>>num;
--front %= size;
s[front]=num;
cout<<"the number is inserted into the queue\n";
}
}
break;
}
case 2:
{
cout<<"rear enqueue\n";
if(front == -1)
{
front=0;
rear=0;
cout<<"enter the element you want to push into the queue\n";
cin>>num;
s[rear]=num;
cout<<"the number is inserted into the queue\n";
}
else if((rear+1)%size==front)
```

```
{
cout<<"queue exausted........\n";
}
else
{
cout<<"enter the element you want to push into the queue\n";
cin>>num;
++rear%=size;
s[rear]=num;
cout<<"the number is inserted into the queue\n";
}
break;
}
default:
{
cout<<"entered wrong choice...";
break;
}
}
}
void dequeue()
{
cout<<"you have selected dequeue operation\n";
int c2;
cout<<"from where u wanna delete the element:(1:front 2:rear):-";
cin>>c2;
switch(c2)
{
case 1:
{
cout<<"front dequeue\n";
```

```
if(rear==-1&&front==-1)
{
cout<<"no element present in the stack\n";
}
else
{
if(rear==front)
{
ele=s[front];
cout<<"you have removed "<<ele<<" front the queue....";
rear=-1;
front=-1;
}
else
{
++front%=size;
}
}
break;
}
case 2:
{
cout<<"rear dequeue\n";
if(rear==-1&&front==-1)
{
cout<<"no element present in the stack\n";
}
else
{
if(rear==front)
{
```

```
ele=s[front];
cout<<"you have removed "<<ele<<" front the queue....";
rear=-1;
front=-1;
}
else
{
--rear %=size;
}
}
break;
}
default:
{
cout<<"entered wrong choice...";
break;
}
}
}
void dis()
{
cout<<"display option is selected\n";
if(front==-1)
{
cout<<"no element is present in array\n";
}
else
{
int t=front;
cout << "Queue is ";
do
```

```
{
cout<<s[t]<<" ";
++t%=size;
}
while(t!=(rear+1)%size);
}
}
void get()
{
front=0;
rear=-1;
cout<<"enter the size of the queue:-\n";
cin>>size;
int s[size];
do
{
cout<<"enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-\n";
cin>>choice;
switch(choice)
{
case 1:
enqueue();
break;

case 2:
dequeue();
break;

case 3:
dis();
break;
```

Queue programs

```
case 4:
break;

default:
cout<<"invalid choice ## enter right choice........\n";
}
}
while(choice!=4);
}
};

int main()
{
nick o;
o.get();
}
```

Output:--

Queue programs

```
enter the size of the queue:-
4
enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-
1
enqueue operation is selected
from where u wanna insert the element:(1:front 2:rear):-1
front enqueue
enter the element you want to push into the queue
1
the number is inserted into the queue
enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-
1
enqueue operation is selected
from where u wanna insert the element:(1:front 2:rear):-1
front enqueue
enter the element you want to push into the queue
2
the number is inserted into the queue
enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-
1
enqueue operation is selected
from where u wanna insert the element:(1:front 2:rear):-2
rear enqueue
enter the element you want to push into the queue
3
the number is inserted into the queue
```

```
enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-
1
enqueue operation is selected
from where u wanna insert the element:(1:front 2:rear):-2
rear enqueue
enter the element you want to push into the queue
4
the number is inserted into the queue
enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-
3
display option is selected
Queue is 2 1 3 4 enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-
2
you have selected dequeue operation
from where u wanna delete the element:(1:front 2:rear):-1
front dequeue
enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-
3
display option is selected
Queue is 1 3 4 enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-
2
you have selected dequeue operation
from where u wanna delete the element:(1:front 2:rear):-2
rear dequeue
enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-
3
display option is selected
Queue is 1 3 enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-
```

1.3Circular_Queue_13

Code:

```
#include<iostream>
```

```
using namespace std;
class nick
{
public:
int front,rear,choice,num,size,s[10],ele;
public:
void enqueue()
{
cout<<"enqueue operation is selected\n";
if(front==-1)
{
front=rear=0;
cout<<"enter the element you want to push into the queue\n";
cin>>num;
s[rear]=num;
cout<<"the number is inserted into the queue\n";
}
else if((rear+1)%size==front)
{
cout<<"queue exausted........\n";
}
else
{
cout<<"enter the element you want to push into the queue\n";
cin>>num;
rear=(rear+1)%size;
s[rear]=num;
}
}
void dequeue()
{
```

```
cout<<"you have selected dequeue operation\n";
if(rear==-1)
{
cout<<"no element present in the stack\n";
}
else
{
if(rear==front)
{
ele=s[front];
cout<<"you have removed "<<ele<<" front the queue....";
rear=-1;
front=-1;
}
else
{
ele=s[front];
cout<<"you have removed "<<ele<<" front the queue....";
front=(front+1)%size;
}
}
}

void dis()
{
cout<<"display option is selected\n";
if(front==-1)
{
cout<<"no element is present in array\n";
}
else
```

```
{
for(int i=front;i!=rear;++i %= size)
{
cout<<i<<" : "<<s[i]<<"\n";
}
cout<<rear<< " : "<<s[rear]<<"\n";
}
}

void get()
{
rear=-1;
front=-1;
cout<<"enter the size of the queue:-\n";
cin>>size;
int s[size];
do
{
cout<<"enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-\n";
cin>>choice;
switch(choice)
{
case 1:
enqueue();
break;

case 2:
dequeue();
break;

case 3:
```

```
dis();
break;

case 4:
break;

default:
cout<<"invalid choice ## enter right choice........\n";
}
}
while(choice!=4);
}
};

int main()
{
nick o;
o.get();
return 0;
}
```

Output:--

Queue programs

```
enter the size of the queue:-
3
enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-
1
enqueue operation is selected
enter the element you want to push into the queue
1
the number is inserted into the queue
enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-
1
enqueue operation is selected
enter the element you want to push into the queue
2
enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-
1
enqueue operation is selected
enter the element you want to push into the queue
3
enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-
1
enqueue operation is selected
queue exausted........
enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-
2
you have selected dequeue operation
you have removed 1 front the queue....enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-
1
enqueue operation is selected
enter the element you want to push into the queue
4
```

```
enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-
3
display option is selected
1 : 2
2 : 3
0 : 4
enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-

2
you have selected dequeue operation
you have removed 2 front the queue....enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-
2
you have selected dequeue operation
you have removed 3 front the queue....enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-

2
you have selected dequeue operation
you have removed 4 front the queue....enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-

2
you have selected dequeue operation
no element present in the stack
enter the option you want to select:-(1-enqueue 2-dequeue 3-display 4-exit):-
```

1.4Johnson's_Algorithm_13

Code:-

```
#include<iostream>

using namespace std;

class nick

{

public:

int *a1,*a2,*q,s,pos1=0,pos2=0,small1,small2,i,front,rear,flag,v;
```

```
public:
int loop()
{
flag=0;
for(i=0;i<s;i++)
{
if(a1[i]==1000&&a2[i]==1000)
{
flag=flag+1;
}
}
if(flag==s)
{
v=2;
}
else
{
v=0;
}
}
void get()
{
front=-1;
rear=-1;
cout<<"johnson algorithm\n";
cout<<"enter the number of processes you wannna take :-\n";
cin>>s;
a1=new int[s];
a2=new int[s];
q=new int[s];
for(int i=0;i<s;i++)
```

```
{
cout<<"enter the process time for first system:-\n";
cin>>a1[i];
cout<<"enter the process time for second system:-\n";
cin>>a2[i];
}
cout<<"processes "<<"system 1  "<<"system 2 \n";
for(int i=0;i<s;i++)
{
cout<<"p"<<i<<"\t    "<<a1[i]<<" \t     "<<a2[i]<<endl;
}
}
void mytech()
{
do
{

//smallest in first array
small1=a1[0];
for(i=1;i<s; i++)
{
if(small1>a1[i])
{
small1=a1[i];
pos1=i;
}
}
cout<<small1<<" "<<pos1<<endl;

//smallest in second array
small2=a2[0];
```

Queue programs

```
for(i=1;i<s; i++)
{
if(small2>a2[i])
{
small2=a2[i];
pos2=i;
}
}
cout<<small2<<" "<<pos2<<endl;
if(small1>small2)
{
        cout<<"small 1 is greater\n";
}
else if(small1<small2)
{
        cout<<"small 2 is greater\n";
}
else
{
        cout<<"same\n";
}
if(small1<small2)
{
fe(pos1);
a1[pos1]=1000;
a2[pos1]=1000;
}
else if(small1>small2)
{
re(pos2);
a1[pos2]=1000;
```

Queue programs

```
a2[pos2]=1000;
}
else
{
fe(pos1);
a1[pos1]=1000;
a2[pos1]=1000;
re(pos2);
a1[pos2]=1000;
a2[pos2]=1000;
}
for(i=0;i<s;i++)
{
cout<<"p"<<i<<"     "<<q[i]<<endl;
}
for(i=0;i<s;i++)
{
cout<<i<<"     "<<a1[i]<<"  "<<a2[i]<<endl;
}
loop();
}
while(v!=2);
cout<<i-1<<"     "<<q[0]<<"   "<<q[s-1]<<endl;
}
void fe(int pos)
{
if(front == -1){
front = rear = 0;
q[rear] = pos;
}
else
```

```
{
if(front==0)
{
front = s;
}
if((front-1)%s==rear)
{
cout << "Overflow.\n";
}
else
{
--front %= s;
q[front] = pos;
}
}
}
void re(int pos)
{
if(front == -1)
{
front = rear = 0;
q[rear] = pos;
}
else if((rear+1)%s==front)
{
cout << "Overflow.\n";
}
else
{
++rear %= s;
q[rear] = pos;
```

```
}
}
};
int main()
{
nick o;
o.get();
o.mytech();
}
```

Output:--

```
johnson algorithm
enter the number of processes you wannna take :-
4
enter the process time for first system:-
1
enter the process time for second system:-
2
enter the process time for first system:-
3
enter the process time for second system:-
4
enter the process time for first system:-
5
enter the process time for second system:-
6
enter the process time for first system:-
7
enter the process time for second system:-
8
processes system 1  system 2
p0          1        2
p1          3        4
p2          5        6
p3          7        8
1 0
2 0
small 2 is greater
p0     0
p1     0
p2     1900880
```

Queue programs

```
p0      0
p1      0
p2      1900880
p3      0
0      1000  1000
1      3  4
2      5  6
3      7  8
3 1
4 1
small 2 is greater
p0      0
p1      0
p2      1900880
p3      1
0      1000  1000
1      1000  1000
2      5  6
3      7  8
5 2
6 2
small 2 is greater
p0      0
p1      0
p2      2
p3      1
0      1000  1000
1      1000  1000
2      1000  1000
3      7  8
```

```
3     7  8
5 2
6 2
small 2 is greater
p0     0
p1     0
p2     2
p3     1
0     1000  1000
1     1000  1000
2     1000  1000
3     7  8
7 3
8 3
small 2 is greater
p0     0
p1     3
p2     2
p3     1
0     1000  1000
1     1000  1000
2     1000  1000
3     1000  1000
3     0   1

--------------------------------
Process exited after 31.4 seconds with return value 0
Press any key to continue . . .
```

1.5Round_Robin_13

Code:

```
#include<iostream>

using namespace std;

class nick

{

public:

int d=0,front,rear,size,queue[30],num, readyarr[15], ele,ipinfo[5][7],s[20],timeslice;

public:


void enqueue(int num)


```

```
{
++rear;
queue[rear]=num;
}

int dequeue()
{
if(front>rear)
{
return -1;
}
else
{
ele=queue[front];
front=front+1;
readyarr[d]=ele;
d+=1;
return ele;
}
}
void acceptip()
{
front=0;
rear=-1;
cout<<"Enter number of processes\n";
cin>>size;
cout<<"Enter time slice\n";
cin>>timeslice;
for(int i=0;i<size;i++)
{
cout<<"Enter arrival time and burst time for "<<i<<" processes:\n";
```

```
    for(int j=1;j<3;j++){
        ipinfo[i][0]=i;
        cin>>ipinfo[i][j];
        ipinfo[i][3]=ipinfo[i][2];
        ipinfo[i][4]=-1;
        ipinfo[i][6]=-1;
    }
}
cout<<"Pno. | AT  |  BT  |  RT \n";
for(int i=0;i<size;i++)
{
    for(int j=0;j<4;j++){
       cout<<ipinfo[i][j]<<"   |   ";
    }
    cout<<"\n";
}
init();
}

void init(){
       for(int i=0;i<size;i++){
       if(ipinfo[i][1]==0){
          enqueue(ipinfo[i][0]);

       }
    }

    schedule();
}

void schedule()
```

Queue programs

```
{
   int z=0;
   int i=dequeue();
   while(i!=-1){

      if(ipinfo[i][4]==-1){
         ipinfo[i][4]=z;
      }
      if(ipinfo[i][3]<timeslice){
         for(int j=0;j<ipinfo[i][3];j++){
            s[z]=ipinfo[i][0];



            z+=1;
         ipinfo[i][6]=z;
         }
         ipinfo[i][3]=0;
      }
      else{
          for(int j=0;j<timeslice;j++){
            s[z]=ipinfo[i][0];
            z+=1;
     }
            ipinfo[i][3]=ipinfo[i][3]-timeslice;.
            if(ipinfo[i][3]==0){
               ipinfo[i][6]=z;
                                                        }
            else{
               for(int o=0;o<size;o++){
                  if(ipinfo[o][1]<=z && ipinfo[o][1]>0 && ipinfo[o][3]!=0 && o!=i){
                     enqueue(ipinfo[o][0]);
```

```
                }
            }
                if(ipinfo[i][3]!=0)
                    enqueue(ipinfo[i][0]);
        }

    }
    i=dequeue();
  }
  cout<<"\nProcess queue is: ";
  for(int h=0;h<z;h++){
    cout<<s[h]<<" ";
  }
  cout<<"\n\n";
 cout<<"Pno. | AT  |   BT   |   RT  |  WT   |  TAT  |  CT  |\n";
for(int i=0;i<size;i++)
{
  ipinfo[i][5]=ipinfo[i][6]-ipinfo[i][1];
  ipinfo[i][4]=ipinfo[i][5]-ipinfo[i][2];
  for(int j=0;j<7;j++){
    cout<<ipinfo[i][j]<<"   |   ";
  }
  cout<<"\n";
}

int avgWT=0, avgTAT=0;
for(int i=0;i<size;i++){
  avgWT=ipinfo[i][4]+avgWT;
  avgTAT=ipinfo[i][5]+avgTAT;
}
```

Queue programs

```
cout<<"\nAverage Waiting Time is: "<<avgWT/float(size)<<"\n";
cout<<"\nAverage Turn Around Time is: "<<avgTAT/float(size)<<"\n";
cout<<"\nReady Array is: ";

for(int i=0;i<d;i++){
    cout<<readyarr[i]<<"  ";
}
 cout<<"\n\n";
}
};
int main()
{
nick o;
o.acceptip();
}
```

Output:--

```
Enter number of processes
3
Enter time slice
2
Enter arrival time and burst time for 0 processes:
0
4
Enter arrival time and burst time for 1 processes:
0
3
Enter arrival time and burst time for 2 processes:
0
5
Pno. | AT  |  BT  |  RT
0   |   0   |   4   |   4   |
1   |   0   |   3   |   3   |
2   |   0   |   5   |   5   |

Process queue is: 0 0 1 1 2 2 0 0 1 2 2 2

Pno. | AT  |   BT   |   RT  |  WT   |  TAT  |  CT  |
0   |   0   |   4   |   0   |   4   |   8   |   8   |
1   |   0   |   3   |   0   |   6   |   9   |   9   |
2   |   0   |   5   |   0   |   7   |   12   |   12   |

Average Waiting Time is: 5.66667

Average Turn Around Time is: 9.66667

Ready Array is: 0  1  2  0  1  2  2
```

1.6Queue_Link_List_13

Code:-

```
#include<iostream>

#include<malloc.h>

using namespace std;


struct node{

   int data;

   struct node *next;

}*front=NULL, *rear=NULL, *p, *q, *r, *s;
```

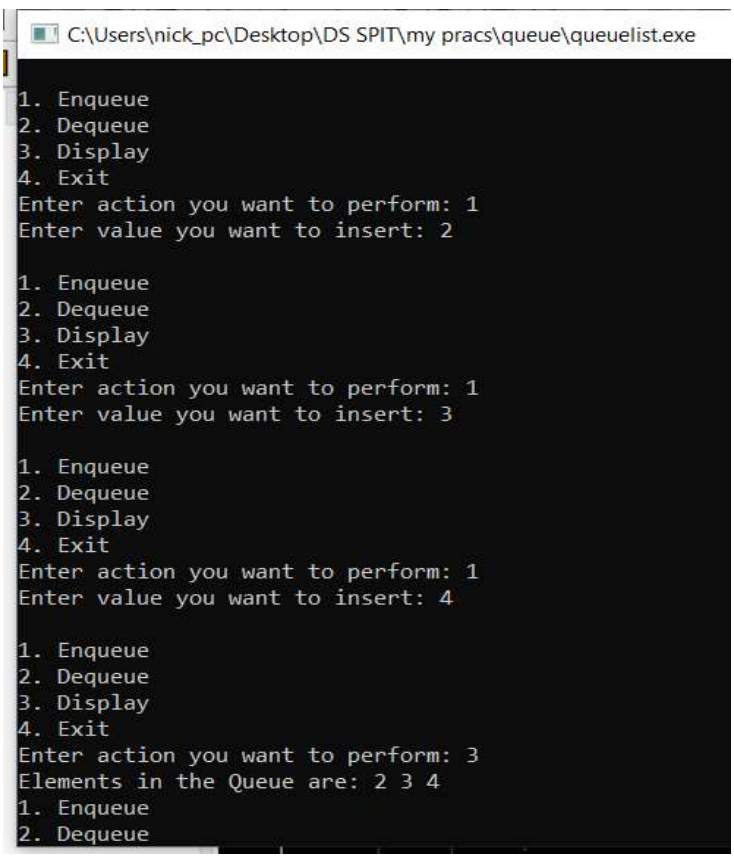
Queue programs

```
class nick{
   int action, value;

   public:
   mytech(){
      do{
         cout << "\n1. Enqueue\n2. Dequeue\n3. Display\n4. Exit\nEnter action you want to perform: ";
         cin >> action;
         switch (action)
         {
         case 1:
            enqueue();
            break;

         case 2:
            dequeue();
            break;

         case 3:
            display();
            break;

         case 4:
            break;

         default:
            cout << "Invalid input." << endl;
            break;
         }
      }while(action != 4);
   }
   void enqueue(){
      cout << "Enter value you want to insert: ";
```

```
    cin >> value;
    p = (struct node*)malloc(sizeof(node));
    p->data = value;
    p->next = NULL;
    if(front==NULL){
        front = p;
        front = rear = p;
    }
    else{
        q = front;
        while(q->next!=NULL)
            q = q->next;
        q->next = p;
        rear = p;
    }
}

void dequeue(){
    if(front==NULL)
        cout << "Underflow." << endl;
    else{
        cout << front->data << " has been removed." << endl;
        front = front->next;
    }
}

void display(){
    if(front==NULL)
        cout << "No Element in the Queue.";
    else{
        p = front;
        cout << "Elements in the Queue are: ";
        while(p!=NULL){
```

```
            cout << p->data << " ";
            p = p->next;
        }
    }
  }
};

int main(){
  nick o;
  o.mytech();
}
```

Output:-



C:\Users\nick_pc\Desktop\DS SPIT\my pracs\queue\queuelist.exe

```
Enter action you want to perform: 2
3 has been removed.

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter action you want to perform: 2
4 has been removed.

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter action you want to perform: 2
Underflow.

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter action you want to perform: 3
No Element in the Queue.
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter action you want to perform: 4
```