Program 1(A) & 1(B)

Name	YOUR NAME
Enrolment Number	YOUR ENROLLMENT NUMBER
Date	

OBJECTIVE

Write a Program to implement following operations in Array:

A. Insertion of an element
B. Deletion of an Element

EQUIPMENT

S.No	Hardware	Software
1.	Intel Core i7	Operating System(Windows 10)
2.	8 GB RAM	Turbo C/C++
3.	Keyboard	MS Word to prepare write up
4.	Mouse	Snipping Tool
5.	Monitor	-
6.	Printer	-

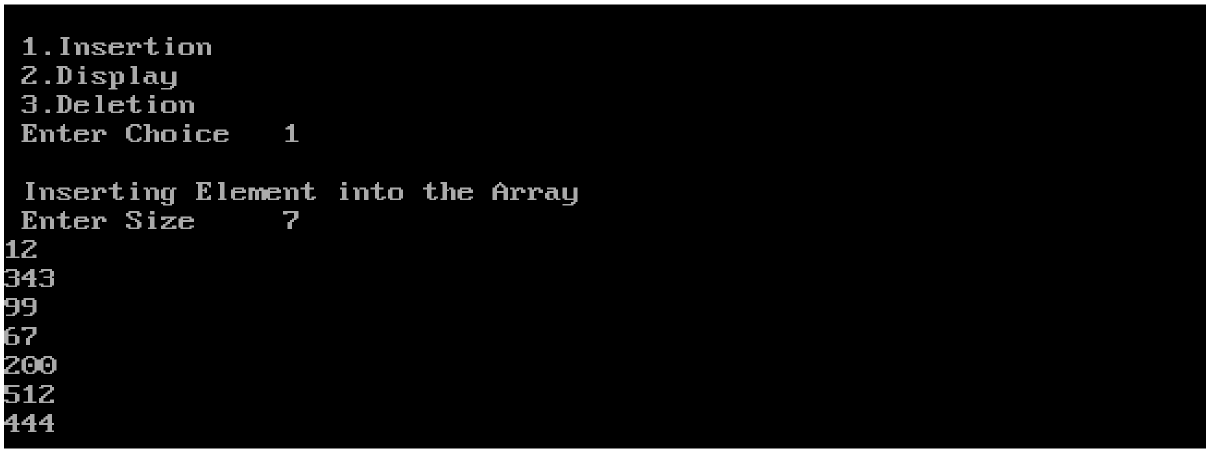
THEORY

An Array is a linear data structure. It is often defined as a continuous memory block containing the same type of data elements. Memory is allocated to an array as a continuous block and entire data of the array is stored together. The size of the array depends on its data type.

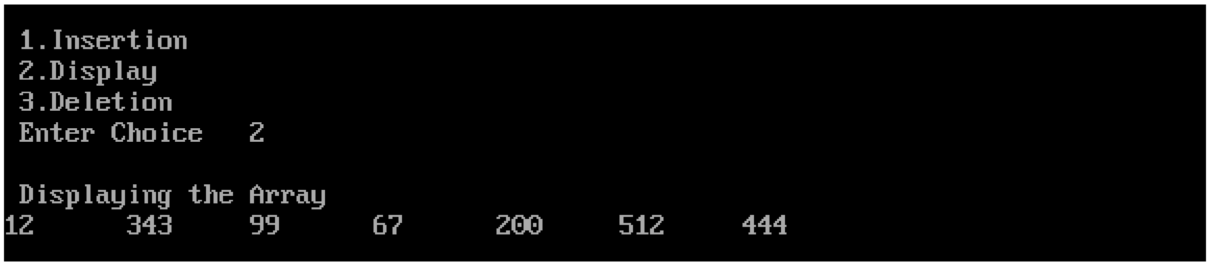
PROGRAM

```
#include<stdio.h>

#include<conio.h>

#include<process.h>
void main()
{
intn,size,i,a[20],b[20],item,j;
clrscr();
do
{
printf("\n 1.Insertion");
printf("\n 2.Display");
printf("\n 3.Deletion");
printf("\n Enter Choice \t");
scanf("%d",&n);
switch(n)
{
case 1: printf("\n Inserting Element into the Array");
        printf("\n Enter Size\t");
        scanf("%d",&size);
        for(i=0; i<size; i++)
        {
        scanf("%d",&a[i]);
        }
        break;

case 2: printf("\n Displaying the Array \n");
        for(i=0; i<size; i++)
        {
        printf("%d \t",a[i]);
        }
        break;

case 3: printf("\n Deleting an Element from the Array");
        printf("\n Enter Element to be Deleted \t");
        scanf("%d",&item);
        for(i=0,j=0; i<size,j<size; i++)
        {
                if(a[i]!=item)
                {
                b[j]=a[i];
                j++;
                }
        }
        printf("\n Displaying the Array \n");
        for(i=0; i<size-1; i++)
        {
        printf("%d \t",b[i]);
        }

        break;
default: exit(0);
}
}while(n!=4);
getch();
}
```

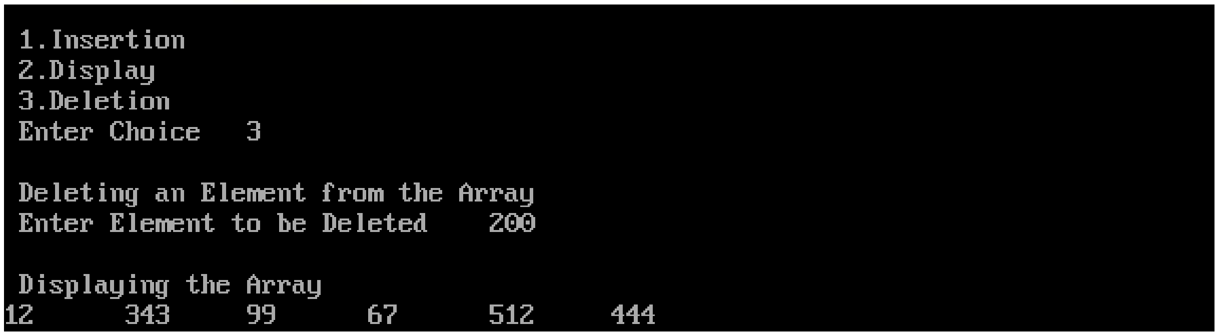
OUTPUT



```
 1.Insertion
 2.Display
 3.Deletion
 Enter Choice   1

 Inserting Element into the Array
 Enter Size     7
12
343
99
67
200
512
444
```



```
 1.Insertion
 2.Display
 3.Deletion
 Enter Choice   2

 Displaying the Array
12      343     99      67      200     512     444
```



```
 1.Insertion
 2.Display
 3.Deletion
 Enter Choice   3

 Deleting an Element from the Array
 Enter Element to be Deleted    200

 Displaying the Array
12      343     99      67      512     444
```

OBSERVATION

The output from the above code can be shown using the above snapshot.

RESULT

The program is successfully written and executed in C language.

Faculty Comments:

Signature:

Faculty: YOUR FACULTY'S NAME
Date:

Program 1(C)

Name	YOUR NAME
Enrolment Number	YOUR ENROLLMENT NUMBER
Date	

OBJECTIVE

Write A Program to implement Linear Search.

EQUIPMENT

S.No	Hardware	Software
1.	Intel Core i7	Operating System(Windows 10)
2.	8 GB RAM	Turbo C/C++
3.	Keyboard	MS Word to prepare write up
4.	Mouse	Snipping Tool
5.	Monitor	-
6.	Printer	-

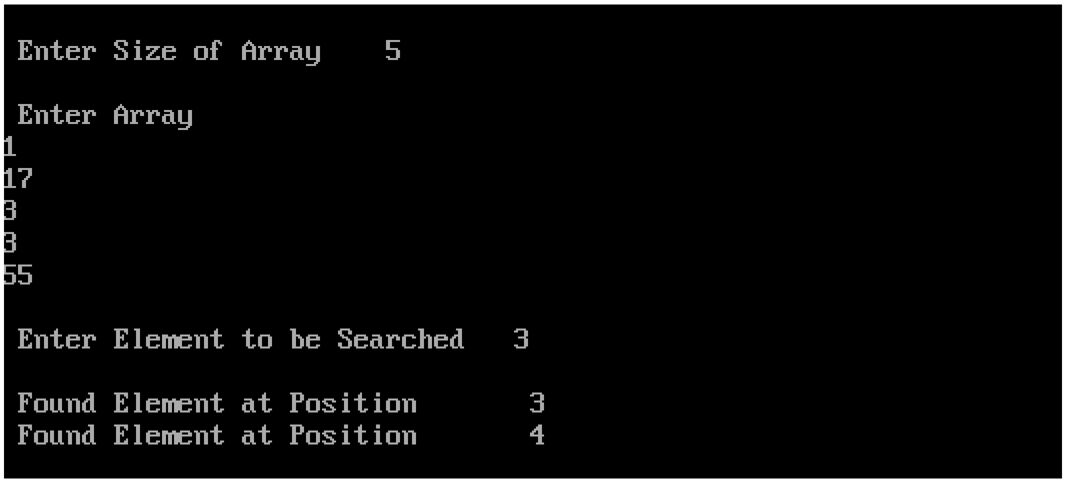
THEORY

Linear Search is a method of searching for an element in an Array. The principle of Linear Search is to compare each and every element in the Array with the desired element to be found, once the match is found the desired output is given. The entire array is traversed to find the element that is to be searched.

PROGRAM

```
#include<stdio.h>
#include<conio.h>

void main()
{
intsize,a[20],i,item;
clrscr();
printf("\n Enter Size of Array \t");
scanf("%d",&size);
printf("\n Enter Array \n");
for(i=0; i<size; i++)
{
scanf("%d",&a[i]);
}

printf("\n Enter Element to be Searched \t");
scanf("%d",&item);

for(i=0; i<size; i++)
{
        if(a[i]==item)
        {
        printf("\n Found Element at Position \t %d",i+1);
        }
}
getch();
}
```

OUTPUT



```
 Enter Size of Array    5

 Enter Array
1
17
3
3
55

 Enter Element to be Searched   3

 Found Element at Position       3
 Found Element at Position       4
```

OBSERVATION

The output from the above code can be shown using the above snapshot.

RESULT

The program is successfully written and executed in C language.

Faculty Comments:

Signature:

Faculty: YOUR FACULTY'S NAME
Date:

Program 1(D)

Name	YOUR NAME
Enrolment Number	YOUR ENROLLMENT NUMBER
Date	

OBJECTIVE

Write A Program to implement Binary Search.

EQUIPMENT

S.No	Hardware	Software
1.	Intel Core i7	Operating System(Windows 10)
2.	8 GB RAM	Turbo C/C++
3.	Keyboard	MS Word to prepare write up
4.	Mouse	Snipping Tool
5.	Monitor	-
6.	Printer	-

THEORY

Binary Search is a method for searching an element within an Array. The principle of Binary Search is to locate the middle element of the Array and search for the required element with respect to this middle element. If the element is greater than the middle element of the array, then the middle element is taken as the beginning whereas if it is less than the middle element of the array, then the middle element is taken as the end of the array.

The Limitations of Binary Search are:

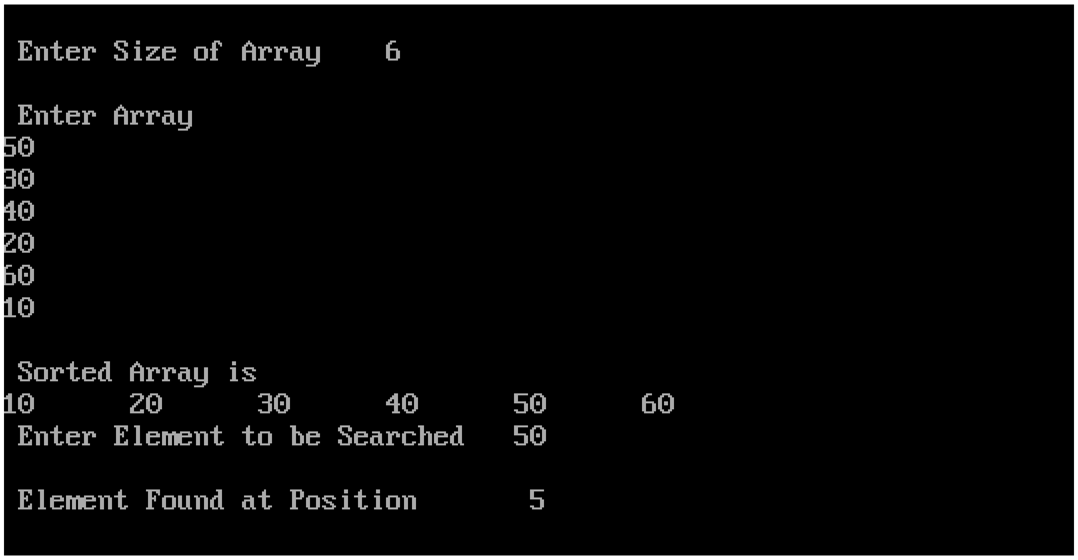
1.The array must be sorted.
2.One must have access to the middle element of the array in any sublist.

PROGRAM

```
#include<stdio.h>
#include<conio.h>
void main()
{
Int beg,mid,end,size,i,a[20],item,j,temp;
clrscr();
printf("\n Enter Size of Array \t");
scanf("%d",&size);
printf("\n Enter Array \n");
for(i=0; i<size; i++)
{
scanf("%d",&a[i]);
}
for(i=0; i<size; i++)
{
        for(j=0; j<size; j++)
        {
                if(a[j]>a[j+1])
                {
                temp=a[j];
                a[j]=a[j+1];
                a[j+1]=temp;
                }
        }
}
printf("\n Sorted Array is \n");
for(i=0; i<size; i++)
{
printf("%d \t",a[i]);
}

printf("\n Enter Element to be Searched \t");
scanf("%d",&item);

beg=0;
end=size;
mid=(beg+end)/2;
while(beg<=end)
{
        if(a[mid]>item)
```

```
            end=mid-1;
            else
            beg=mid+1;
mid=(beg+end)/2;
}
if(a[mid]==item)
printf("\n Element Found at Position\t %d",mid+1);

  else
printf("\n Element Not Found");

getch();
}
```

OUTPUT



```
 Enter Size of Array    6

 Enter Array
50
30
40
20
60
10

 Sorted Array is
10      20      30      40      50      60
 Enter Element to be Searched   50

 Element Found at Position       5
```

OBSERVATION
The output from the above code can be shown using the above snapshot.

RESULT
The program is successfully written and executed in C language.

Faculty Comments:

Signature:

Faculty: YOUR FACULTY’S NAME
Date:

Program 2(A)

Name	YOUR NAME
Enrollment number	YOUR ENROLLMENT NUMBER
Date	

Objective: Write a Program to implement Bubble sorting technique.

Equipment:

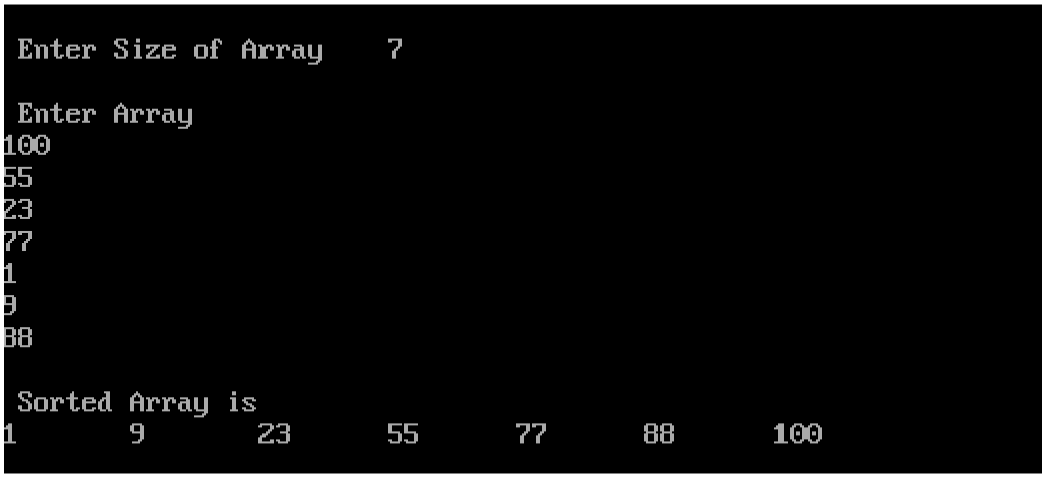
S.No.	**Hardware**	**Software**
1.	Intel core i5	Operating System(Windows 10)
2.	4GB RAM	Turbo C/C++
3.	Keyboard	MS Word to prepare write up
4.	Mouse	Snipping tool
5.	Monitor	-
6.	Printer	-

Theory : Bubble Sort is a simple sorting algorithm that works by repeatedly stepping through the list to be sorted, comparing each pair of adjacent items and swapping them if they are in the wrong order.

Programming:

```
#include<stdio.h>
#include<conio.h>

void main()
{
inta[20],i,j,size,temp;
clrscr();
printf("\n Enter Size of Array \t");
scanf("%d",&size);
printf("\n Enter Array \n");
for(i=0; i<size; i++)
{
scanf("%d",&a[i]);
}

for(i=0; i<size; i++)
{
        for(j=0; j<size; j++)
        {
                if(a[j]>a[j+1])
                {
                temp=a[j];
                a[j]=a[j+1];
                a[j+1]=temp;
                }
        }
}
printf("\n Sorted Array is \n");
for(i=0; i<size; i++)
{
printf("%d \t",a[i]);
}

getch();
}
```

OUTPUT



```
 Enter Size of Array    7

 Enter Array
100
55
23
77
1
9
88

 Sorted Array is
1       9       23      55      77      88      100
```

OBSERVATION
The output from the above code can be shown using the above snapshot.

RESULT
The program is successfully written and executed in C language.

Faculty Comments:

Signature:

Faculty: YOUR FACULTY'S NAME
Date:

Program 2(B)

Name	YOUR NAME
Enrollment number	YOUR ENROLLMENT NUMBER
Date	

Objective: Write a Program to implement Insertion sorting technique.

Equipment:

S.No.	**Hardware**	**Software**
1.	**Intel core i5**	**Operating System(Windows 10)**
2.	**4GB RAM**	**Turbo C/C++**
3.	**Keyboard**	**MS Word to prepare write up**
4.	**Mouse**	**Snipping tool**
5.	**Monitor**	**-**
6.	**Printer**	**-**

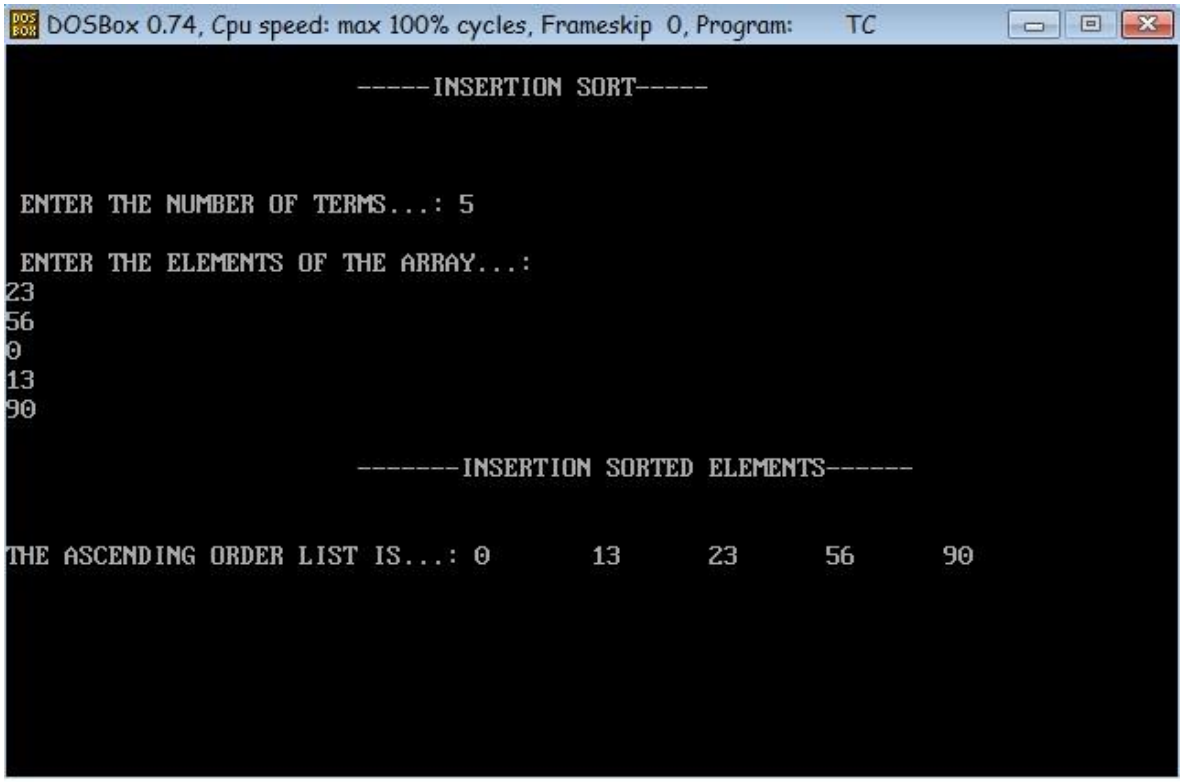
Theory : Insertion sort is a simple sorting algorithm that builds the final sorted array one item at a time. It is much less efficient on large lists than more advanced algorithms such as quicksort, heapsort, or merge sort.

Programming:

```
#include<stdio.h>

#include<conio.h>

void main()

{
```

```
{
int A[20],n,Temp,i,j;
clrscr();
printf("\n\t\t\t-----INSERTION SORT-----\n\n");
printf("\n\n ENTER THE NUMBER OF TERMS...: ");
scanf("%d",&n);
printf("\n ENTER THE ELEMENTS OF THE ARRAY...:\n");
for(i=0; i < n;i++)
{
        scanf("%d", &A[i]);
}
for(i=1; i< n; i++)
{
        Temp = A[i];
        j = i-1;
        while(Temp < A[j] && j>=0)
        {
                A[j+1] = A[j];
                j = j-1;
        }
        A[j+1] = Temp;
}
printf("\n\t\t\t-------INSERTION SORTED ELEMENTS------\n\n");
printf("\nTHE ASCENDING ORDER LIST IS...:");
for(i=0; i < n; i++)
{
```

```
                printf("\t%d", A[i]);
        }
        getch();
}
```


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
                         -----INSERTION SORT-----


 ENTER THE NUMBER OF TERMS...: 5

 ENTER THE ELEMENTS OF THE ARRAY...:
23
56
0
13
90

                         -------INSERTION SORTED ELEMENTS------


THE ASCENDING ORDER LIST IS...: 0       13      23      56      90
```

Observation and Discussion :

The output can be seen in the above snapshot.

Result :

The program is successfully written and executed in C language.

Faculty Comments:

Faculty Name: YOUR FACULTY'S NAME

Signature:

Date:

Program 2(C)

Name	YOUR NAME
Enrollment number	YOUR ENROLLMENT NUMBER
Date	

Objective : Write a program to implement Selection Sorting Technique.

Equipment:

S.No.	**Hardware**	**Software**
1.	Intel core i5	Operating System(Windows 10)
2.	4GB RAM	Turbo C/C++
3.	Keyboard	MS Word to prepare write up
4.	Mouse	Snipping tool
5.	Monitor	-
6.	Printer	-

Theory:

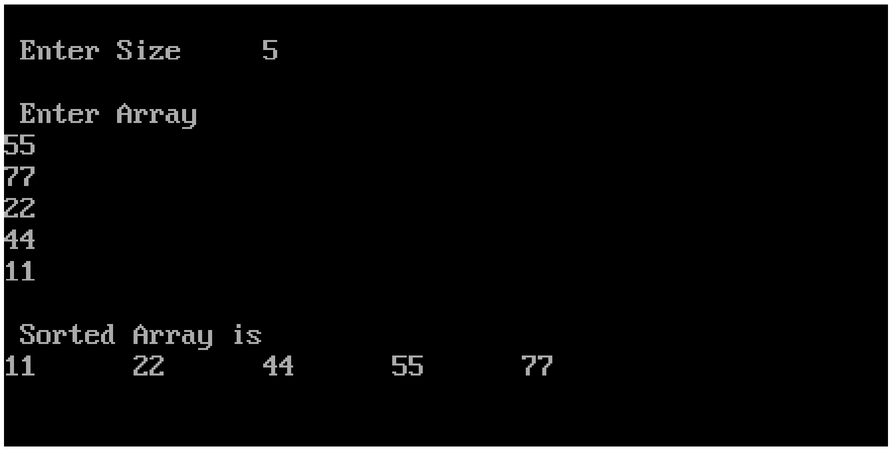
Selection Sort is another sorting technique used to sort an array. The principle of selection sort if to select the smallest element within the unsorted array and progressively sort the array by comparing the rest of the array with this smallest element. Once an element is sorted and allocated to its proper place, it is not included in the next sub-array that is taken into consideration.

Programming:

```
#include<stdio.h>
#include<conio.h>

void main()
{
int size, a[20],i,j,temp;
clrscr();
printf("\n Enter Size \t");
scanf("%d",&size);
printf("\n Enter Array \n");
for(i=0; i<size; i++)
{
scanf("%d",&a[i]);
}

for(i=0; i<size; i++)
{
        for(j=i+1; j<=size; j++)
        {
                if(a[i]>a[j])
                {
                temp=a[j];
                a[j]=a[i];
                a[i]=temp;
                }
        }
}

printf("\n Sorted Array is \n");
for(i=0; i<size; i++)
{
```

```
printf("%d \t",a[i]);
}
getch();
}
```

OUTPUT


```
 Enter Size     5

 Enter Array
55
77
22
44
11

 Sorted Array is
11      22      44      55      77
```

OBSERVATION
The output from the above code can be shown using the above snapshot.

RESULT
The program is successfully written and executed in C language.

Faculty Comments:

Signature:

Faculty Name: YOUR FACULTY'S NAME
Date:

Program 2(D)

Name	YOUR NAME
Enrollment number	YOUR ENROLLMENT NUMBER
Date	

Objective : Write a Program to implement Quick sorting technique.

Equipment:

S.No.	Hardware	Software
1.	Intel core i5	Operating System(Windows 10)
2.	4GB RAM	Turbo C/C++
3.	Keyboard	MS Word to prepare write up
4.	Mouse	Snipping tool
5.	Monitor	-
6.	Printer	-

Theory: Quicksort is a simple sorting algorithm using the divide-and-conquer recursive procedure. It is the quickest comparison-based sorting algorithm in practice with an average running time of O(n log(n)).It is also known as partition-exchange sort.

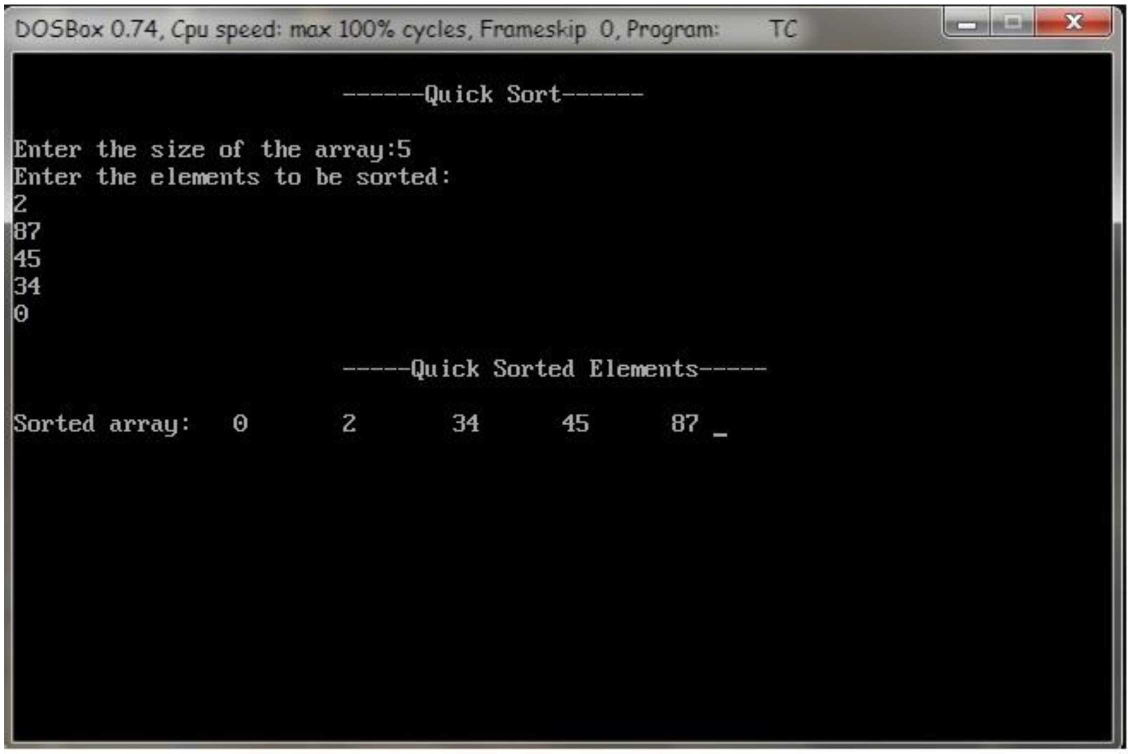
Programming:

```
#include<stdio.h>

#include<conio.h>

void quicksort(int arr[], int lb, int ub);
```

```
   void main()

    {

     int arr[20], n, i;

     clrscr();

     printf("\n\t\t\t------Quick Sort------\n\n");

     printf("Enter the size of the array:");

     scanf("%d",&n);

     printf("Enter the elements to be sorted:\n");

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

     scanf("%d",&arr[i]);

     quicksort(arr, 0, n-1);

     printf("\n\t\t\t-----Quick Sorted Elements-----\n\n");

     printf("Sorted array:");

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

     printf("\t%d ",arr[i]);

     getch();

 }

 void quicksort(int arr[], int lb, int ub)

{

 int pivot, i, j, temp;

 if(lb < ub)
```

```
{
        pivot = lb;
        i = lb;
        j = ub;
        while(i < j)
        {
                while(arr[i] <= arr[pivot] &&  i <= ub)
                i++;
                while(arr[j] > arr[pivot] && j >= lb)
                j--;
                if(i < j)
                {
                        temp = arr[i];
                        arr[i] = arr[j];
         arr[j] = temp;
                }
        }

        temp = arr[j];
        arr[j] = arr[pivot];
  arr[pivot] = temp;
```

```
            quicksort(arr, lb, j-1);

        quicksort(arr, j+1, ub);

       }

}
```



```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip  0, Program:     TC

                       ------Quick Sort------

Enter the size of the array:5
Enter the elements to be sorted:
2
87
45
34
0

                       -----Quick Sorted Elements-----

Sorted array:   0       2       34      45      87 _
```

Observation and Discussion:

The output can be seen in the above snapshot.

Result:

The program is successfully written and executed in C language.

Faculty Comments:

Faculty Name: YOUR FACULTY’S NAME

Signature:

Date:

Program 3

Name	YOUR NAME
Enrolment Number	YOUR ENROLLMENT NUMBER
Date	

OBJECTIVE

WAP to implement Operations in Stack

A. Push operation
B. Pop operation
C. Display operation

EQUIPMENT

S.No	Hardware	Software
1.	Intel Core i7	Operating System(Windows 10)
2.	8 GB RAM	Turbo C/C++
3.	Keyboard	MS Word to prepare write up

4.	Mouse	Snipping Tool
5.	Monitor	-
6.	Printer	-

THEORY

A Stack is a list of elements in which an element may be inserted or deleted from only one end of the stack, called Top of the stack. Elements are removed from the Stack in the reverse order of which they were inserted into the stack. It follows Last In First Out approach (LIFO).

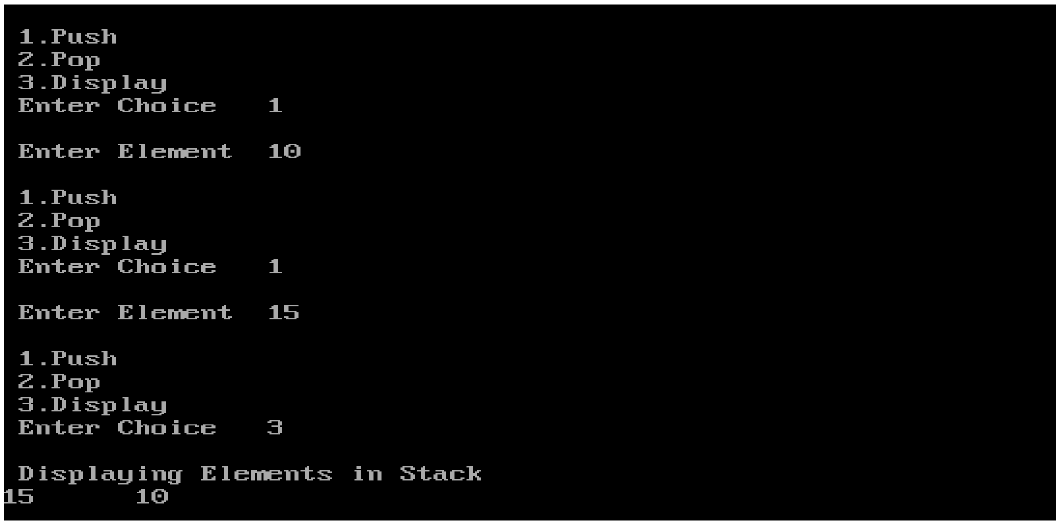
- Push: Inserting an Element into the Stack.
- Pop: Deleting an Element from the Stack.

PROGRAM

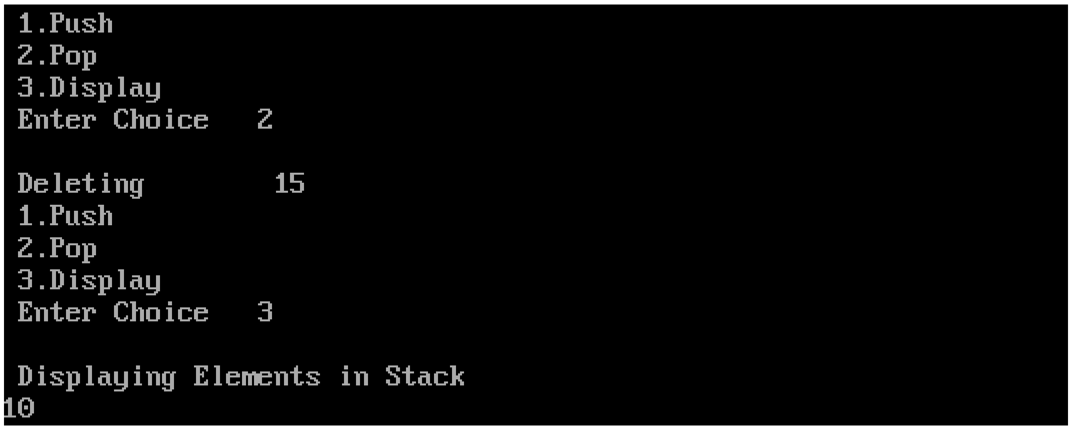
```
#include<stdio.h>
#include<conio.h>

void main()
{
intn,a[6],top=-1,temp;
clrscr();
do
{
printf("\n 1.Push");
printf("\n 2.Pop");
printf("\n 3.Display");
printf("\n Enter Choice \t");
scanf("%d",&n);
switch(n)
{
case 1: if(top==5)
        printf("\n OverFlow");
        else
        {
        top++;
        printf("\n Enter Element \t");
        scanf("%d",&a[top]);
        }
        break;
case 2: if(top==-1)
        printf("\n Underflow \t");
        else
        {
        printf("\n Deleting \t %d",a[top]);
        top--;
        }
        break;
case 3: printf("\n Displaying Elements in Stack \n");
        temp=top;
        while(temp!=-1)
        {
        printf("%d \t",a[temp]);
        temp--;
        }
```

```
            break;
default: exit(0);
}
}while(n!=4);
getch();
}
```

OUTPUT



```
 1.Push
 2.Pop
 3.Display
 Enter Choice   1

 Enter Element  10

 1.Push
 2.Pop
 3.Display
 Enter Choice   1

 Enter Element  15

 1.Push
 2.Pop
 3.Display
 Enter Choice   3

 Displaying Elements in Stack
15      10
```



```
 1.Push
 2.Pop
 3.Display
 Enter Choice   2

 Deleting        15
 1.Push
 2.Pop
 3.Display
 Enter Choice   3

 Displaying Elements in Stack
10
```

OBSERVATION
The output from the above code can be shown using the above snapshot.

RESULT
The program is successfully written and executed in C language.

Faculty Comments:

Signature:

Faculty: YOUR FACULTY’S NAME
Date:

Program 4

Name	YOUR NAME
Enrollment number	YOUR ENROLLMENT NUMBER
Date	

Objective : Write a program to implement all the operation in queue

A. Insertion Operation
B. Deletion operation
C. Display operation

Theory: Queues is a kind of abstract data type where items are inserted one end (rear end) known as **enqueue** operation and deteted from the other end(front end) known as **dequeue** operation.
This makes the queue a **First-In-First-Out (FIFO)** data structure.
The queue performs the function of a buffer.

Programming:

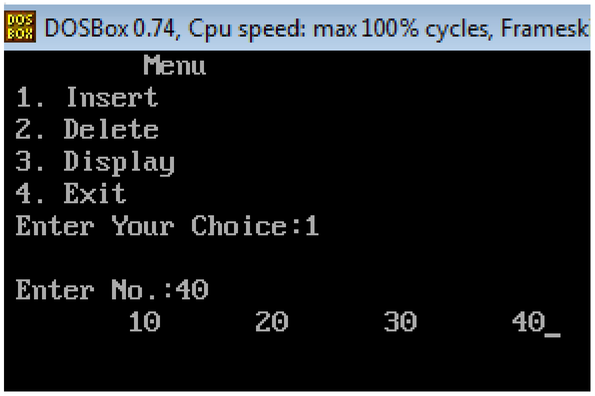
```
#include<stdio.h>

#include<conio.h>

#define SIZE 5

int front=-1;
int rear=-1;

int q[SIZE];

void insert();

void del();

void display();

void main()

{

        int choice;

        clrscr();

        do

        {

                clrscr();

                printf("\t Menu");

                printf("\n 1. Insert");

                printf("\n 2. Delete");

                printf("\n 3. Display ");

                printf("\n 4. Exit");

                printf("\n Enter Your Choice:");

                scanf("%d",&choice);

                switch(choice)

                {
```

```
                case 1:
                        insert();
                        display();
                        getch();
                        break;
                case 2:
                         del();
                         display();
                         getch();
                         break;
                case 3:
                        display();
                        getch();
                        break;
                case 4:
                        printf("End of Program....!!!!");
                        getch();
                        exit(0);
            }
        }while(choice!=4);
}
```

```
void insert()
{
        int no;
        printf("\n Enter No.:");
        scanf("%d",&no);
        if(rear < SIZE-1)
        {
                q[++rear]=no;
                if(front==-1)
                front=0;// front=front+1;
        }
        else
        {
                printf("\n Queue overflow");
        }
}
void del()
{
        if(front==-1)
        {
                printf("\nQueue Underflow");
```

```
                        return;
            }
                else
                {
                        printf("\nDeleted Item:-->%d\n",q[front]);
                }
                if(front==rear)
                {
                        front=-1;
                        rear=-1;
                }
                else
                {
                        front=front+1;
                }
}
 void display()
{
                int i;
                if(front==-1)
                {
```

```
                printf("\nQueue is empty....");

                return;

        }

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

                printf("\t%d",q[i]);

}
```



```
DOSBox 0.74, Cpu speed: max 100% cycles, Framesk
         Menu
1.  Insert
2.  Delete
3.  Display
4.  Exit
Enter Your Choice:1

Enter No.:40
        10      20      30      40_
```

OBSERVATION

The output from the above code can be shown using the above snapshot.

RESULT

The program is successfully written and executed in C language.

Faculty Comments:

Signature:

Faculty: YOUR FACULTY'S NAME
Date:

Program 5

Name	YOUR NAME
Enrolment Number	YOUR ENROLLMENT NUMBER
Date	

OBJECTIVE

WAP to implement all the operations in Linked stack

(a). Push operation

(b). Pop operation

(c). Display operation

EQUIPMENT

S.No	Hardware	Software
1.	Intel Core i7	Operating System(Windows 10)
2.	8 GB RAM	Turbo C/C++
3.	Keyboard	MS Word to prepare write up
4.	Mouse	Snipping Tool
5.	Monitor	-
6.	Printer	-

THEORY

A stack is a list of elements in which an element may be inserted or deleted only at one end, called the top of the stack. The linked representation of stack also called linked stack is a stack that is implemented using a singly linked list.
There are basically three operations that can be performed on linked stacks.

1) PUSH: Insert an element into a stack
2) POP: Delete an element from the stack
3) Display: Display the elements in the stack

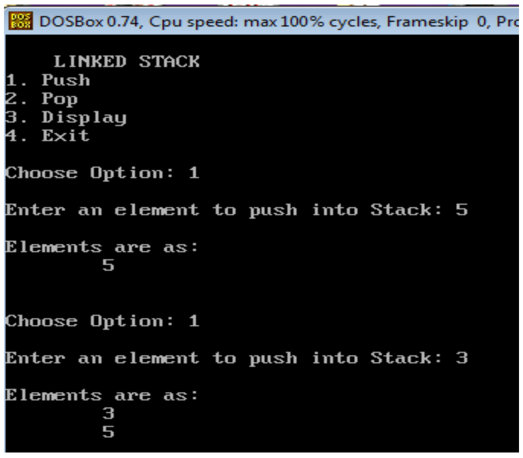
PROGRAM

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
struct Node
{
   int Data;
   struct Node *next;
}*top;
void popStack()
{
   struct Node *temp, *var=top;
   if(var==top)
   {
        top = top->next;
   }
   else
   printf("\nUnderflow. Stack is Empty");
}
void push(int value)
{
   struct Node *temp;
   temp=(struct Node *)malloc(sizeof(struct Node));
   temp->Data=value;
   if (top == NULL)
   {
         top=temp;
         top->next=NULL;
   }
   else
   {
        temp->next=top;
        top=temp;
   }
}
void display()
{
    struct Node *var=top;
    if(var!=NULL)
    {
          printf("\nElements are as:\n");
```

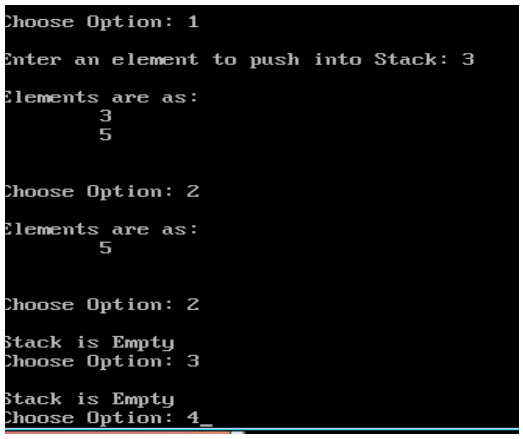
```
            while(var!=NULL)
            {
                printf("\t%d\n",var->Data);
                var=var->next;
            }
     printf("\n");
     }
     else
     printf("\nStack is Empty");
}
void main()
{
int i=0;
clrscr();
top=NULL;
printf("\n    LINKED STACK");
printf(" \n1. Push");
printf(" \n2. Pop ");
printf(" \n3. Display");
printf(" \n4. Exit\n");
while(1)
          {
          printf(" \nChoose Option: ");
          scanf("%d",&i);
          switch(i)
            {
            case 1: {
                      int value;
                      printf("\nEnter an element to push into Stack: ");
                      scanf("%d",&value);
                      push(value);
                      display();
                      break;
                      }
               case 2:  {
                      popStack();
                      display();
                      break;
                      }
               case 3:  {
                      display();
```

```
                        break;
                        }
                case 4:  exit(0);
                default: printf("\nwrong choice entered");
              }
        }
}
```

OUTPUT



```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip  0, Pro

    LINKED STACK
1. Push
2. Pop
3. Display
4. Exit

Choose Option: 1

Enter an element to push into Stack: 5

Elements are as:
        5


Choose Option: 1

Enter an element to push into Stack: 3

Elements are as:
        3
        5
```



```
hoose Option: 1

nter an element to push into Stack: 3

lements are as:
        3
        5


hoose Option: 2

lements are as:
        5


hoose Option: 2

tack is Empty
hoose Option: 3

tack is Empty
hoose Option: 4_
```

OBSERVATION

The output from the above code can be shown using the above snapshot.

RESULT

The program is successfully written and executed in C language.

Faculty Comments:

Signature:

Faculty: YOUR FACULTY'S NAME
Date:

Program 6

Name	YOUR NAME
Enrolment Number	YOUR ENROLLMENT NUMBER
Date	

OBJECTIVE

WAP to implement all the operations in Linked Queue

(a). Insertion operation

(b). Deletion operation

(c). Display operation

EQUIPMENT

S.No	Hardware	Software
1.	Intel Core i7	Operating System(Windows 10)
2.	8 GB RAM	Turbo C/C++
3.	Keyboard	MS Word to prepare write up
4.	Mouse	Snipping Tool
5.	Monitor	-
6.	Printer	-

THEORY

A queue is a linear list of elements in which deletions can take place only at one end, called the front, and insertions can take place only at the other end called the rear. A linked queue is a queue implemented as a linked list with two pointer variables FRONT and REAR pointing to the nodes which is in the FRONT and REAR of the queue.

There are basically three operations that can be performed on linked queues.

1) Insertion: Insert an element into a queue

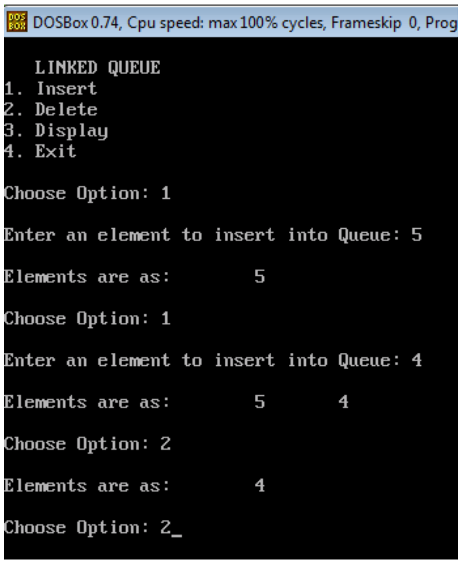
2) Deletion: Delete an element from the queue
3) Display: Display the elements in the queue

PROGRAM

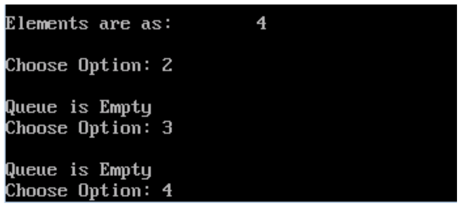
```
#include<stdio.h>
 #include<conio.h>
 #include<stdlib.h>
 struct Node
 {
        int Data;
        struct Node* next;
 }*rear, *front;

void delQueue()
{
       struct Node *temp, *var=rear;
      if(var==rear)
      {
             rear = rear->next;
             free(var);
      }
      else
      printf("\nQueue Empty");
}
void insert(int value)
{
     struct Node *temp;
     temp=(struct Node *)malloc(sizeof(struct Node));
     temp->Data=value;
     if (front == NULL)
     {
           front=temp;
           front->next=NULL;
           rear=front;
     }
     else
     {
           front->next=temp;
           front=temp;
           front->next=NULL;
     }
}
void display()
{
    struct Node *var=rear;
    if(var!=NULL)
    {
        printf("\nElements are as:  ");
        while(var!=NULL)
        {
            printf("\t%d",var->Data);
            var=var->next;
        }
    printf("\n");
    }
    else
    printf("\nQueue is Empty");
}
int main()
{
int i=0;
clrscr();
front=NULL;
printf("\n   LINKED QUEUE");
printf(" \n1. Insert");
printf(" \n2. Delete");
printf(" \n3. Display");
printf(" \n4. Exit\n");
while(1)
      {
      printf(" \nChoose Option: ");
      scanf("%d",&i);
      switch(i)
      {
            case 1:{
                  int value;
                  printf("\nEnter an element to insert into Queue: ");
                  scanf("%d",&value);
                  insert(value);
                  display();
                  break;
                  }
            case 2:{
                  delQueue();
                  display();
```

```
                break;
                }
            case 3:{
                display();
                break;
                }
            case 4:exit(0);
            default:printf("\nwrong choice entered");
               }
        }
}
```

OUTPUT



```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip  0, Prog

   LINKED QUEUE
1. Insert
2. Delete
3. Display
4. Exit

Choose Option: 1

Enter an element to insert into Queue: 5

Elements are as:        5

Choose Option: 1

Enter an element to insert into Queue: 4

Elements are as:        5       4

Choose Option: 2

Elements are as:        4

Choose Option: 2_
```



```
Elements are as:        4

Choose Option: 2

Queue is Empty
Choose Option: 3

Queue is Empty
Choose Option: 4
```

OBSERVATION

The output from the above code can be shown using the above snapshot.

RESULT

The program is successfully written and executed in C language.

Faculty Comments:

Signature:

Faculty: YOUR FACULTY'S NAME
Date: