**Experiment No.4**

**Title:** Implementation of Queue using C

**Problem Statement:**

Write a menu driven C program to implement queue and perform the following operations

(i)Insert

(ii)Remove

(iii)Display

**Algorithm:**

**Step 1 -**Include all the **header files** which are used in the program and define a constant **'SIZE'** with specific value.

**Step 2 -**Declare all the **user defined functions** which are used in queue implementation.

**Step 3 -**Create a one dimensional array with above defined SIZE (**int queue[SIZE]**)

**Step 4 -**Define two integer variables **'front'** and '**rear**' and initialize both with **'-1'**. (**int front = -1, rear = -1**)

Step 5 **-**Then implement main method by displaying menu of operations list and make suitable function calls to perform operation selected by the user on queue.

**insert(value)** - Inserting value into the queue

**Step 1** -Check whether **queue** is **FULL**. (**rear == SIZE-1**)

**Step 2 -**If it is **FULL**, then display **"Queue is FULL!!! Insertion is not possible!!!"** and terminate the function.

**Step 3** -If it is **NOT FULL**, then increment **rear** value by one (**rear++**) and set **queue[rear]** = **value**.

**delete()** - Deleting a value from the Queue

**Step 1 -**Check whether **queue** is **EMPTY**. (**front == rear**)

**Step 2** -If it is **EMPTY**, then display **"Queue is EMPTY!!! Deletion is not possible!!!"** and terminate the function.

**Step 3 -**If it is **NOT EMPTY**, then increment the **front** value by one (**front ++**). Then display **queue[front]** as deleted element. Then check whether both **front** and **rear** are equal (**front** == **rear**), if it **TRUE**, then set both **front** and **rear** to '**-1**' (**front** = **rear** = **-1**).

**display()** - Displays the elements of a Queue

We can use the following steps to display the elements of a queue...

**Step 1 -**Check whether **queue** is **EMPTY**. (**front == rear**)

**Step 2** -If it is **EMPTY**, then display **"Queue is EMPTY!!!"** and terminate the function.

**Step 3 -**If it is **NOT EMPTY**, then define an integer variable '**i**' and set '**i** = **front+1**'.

**Step 4 -**Display '**queue[i]**' value and increment '**i**' value by one (**i++**). Repeat the same until '**i**' value reaches to **rear** (**i** <= **rear**)

**Code:**

#include<stdio.h>

#define SIZE 3

void insert(int);

void delete();

void display();

int queue[SIZE], front = -1, rear = -1;

void main()

{

int value, choice;

while(1){

printf("\n\*\*\*\*\* MENU \*\*\*\*\*\n");

printf("1. Insertion\n2. deletion\n3. Display\n4. Exit");

printf("\nEnter your choice: ");

scanf("%d",&choice);

switch(choice){

case 1: printf("Enter the value to be insert: ");

scanf("%d",&value);

insert(value);

break;

case 2: delete();

break;

case 3: display();

break;

case 4: exit(0);

default: printf("\nWrong selection!!! Try again!!!");

}

}

}

void insert(int value){

if(rear == SIZE-1)

printf("\nQueue is Full!!! Insertion is not possible!!!");

else{

if(front == -1)

front = 0;

rear++;

queue[rear] = value;

}

}

void delete(){

if(front == rear)

printf("\nQueue is Empty!!! Deletion is not possible!!!");

else{

printf("\nDeleted : %d", queue[front]);

front++;

if(front == rear)

front = rear = -1;

}

}

void display(){

if(rear == -1)

printf("\nQueue is Empty!!!");

else{

int i;

printf("\nQueue elements are:\n");

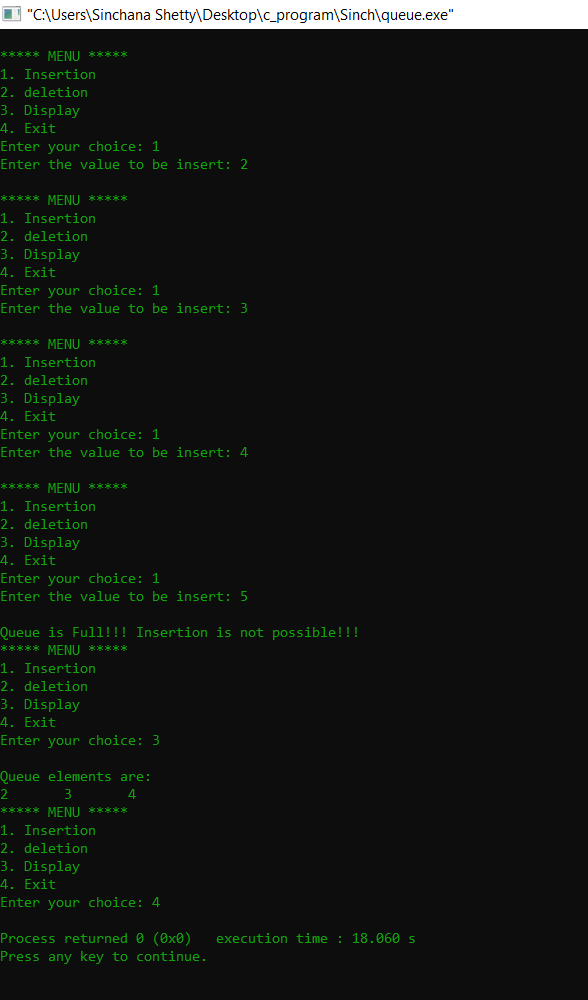
for(i=front; i<=rear; i++)

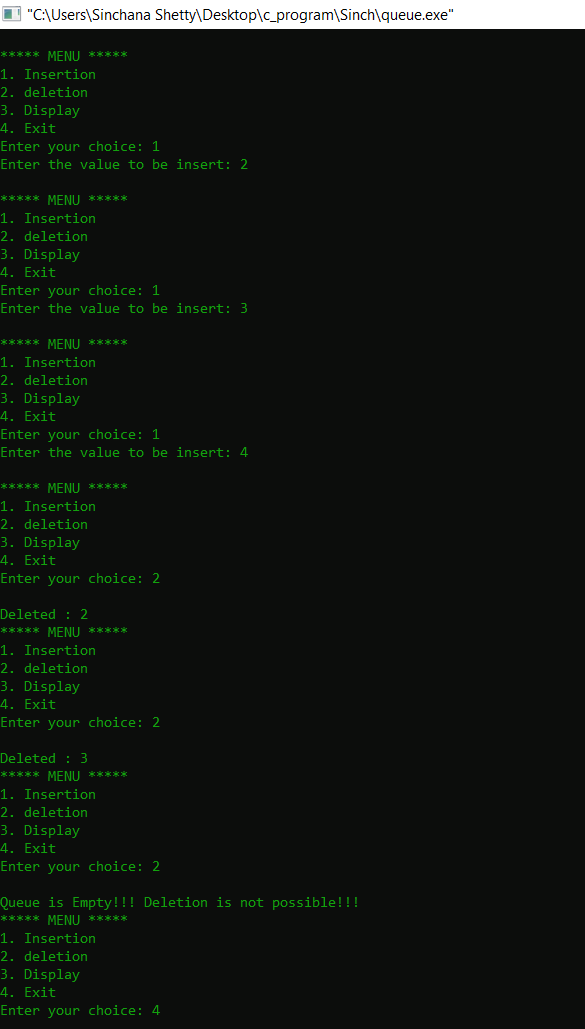
printf("%d\t",queue[i]);

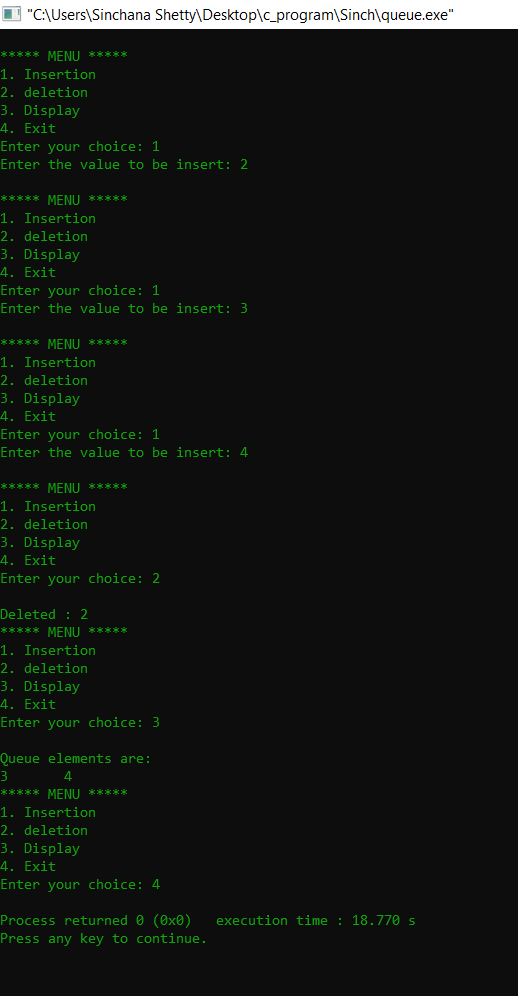
}

}

**Results:**







**Analysis(Limitations):**

One limitation of this implementation is the limited space. The queue will only hold as many or even lesser elements as the array’s size is fixed. Unfilled space will not be utilized as the front pointer of the queue would have moved ahead. One method of overcoming is by using a circular queue, which ensures that the full array is utilized when storing the elements.