**Experiment Number: 4**

**Title:** Implementation of Queue Using C.

**Problem Statement:**

Write a menu driven program in C to perform following operations on the Queue.

1. Insert
2. Delete

**Algorithm:**

Start.

1. Create the Queue by declaring the array of fixed size.
2. Initialize the variables front and rear to -1 which indicate the front end and rear end of the queue respectively.
3. I) For insert operation,

First check if rear has reached to the last of the queue if yes don’t insert the new element, if no, increment the rear variable so that it points to next location of queue and insert the new element.

II) For delete operation,

First check if front variable is -1 or front>rear which indicates all the elements are deleted from the queue,

If yes, don’t delete.

If no, return the element from the front index and increment the front variable so that it points to the next value.

1. Stop.

**Code:**

#include<stdio.h>

#include<stdlib.h>

#define MAX 5

void insert();

void delete();

void display();

int queue\_arr[MAX];

int rear=-1;

int front=-1;

int main()

{

int choice;

while(1)

{

printf("1. Insert an element\t2. Delete an element\t3. Display the queue\t4. Quit");

printf("\nEnter your choice:");

scanf("%d",&choice);

switch(choice)

{

case 1: insert();

break;

case 2: delete();

break;

case 3: display();

break;

case 4: exit(1);

default: printf("\nWrong Choice!!");

break;

}

}

return 0;

}

void insert()

{

int add\_item;

if(rear==MAX-1)

printf("\nQueue Overflow!! Insertion not possible !!\n");

else

{

if(front==-1)

front=0;

printf("\nEnter the element to be inserted in Queue:");

scanf("%d",&add\_item);

rear++;

queue\_arr[rear]=add\_item;

}

}

void delete()

{

if(front==-1||front>rear)

{

printf("Queue Underflow!! Deletion not possible!! \n");

return;

}

else

{

printf("\nElement deleted from Queue is : %d\n",queue\_arr[front]);

front++;

}

}

void display()

{

int i;

if(front==-1)

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

else

{

printf("\nQueue is:\n");

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

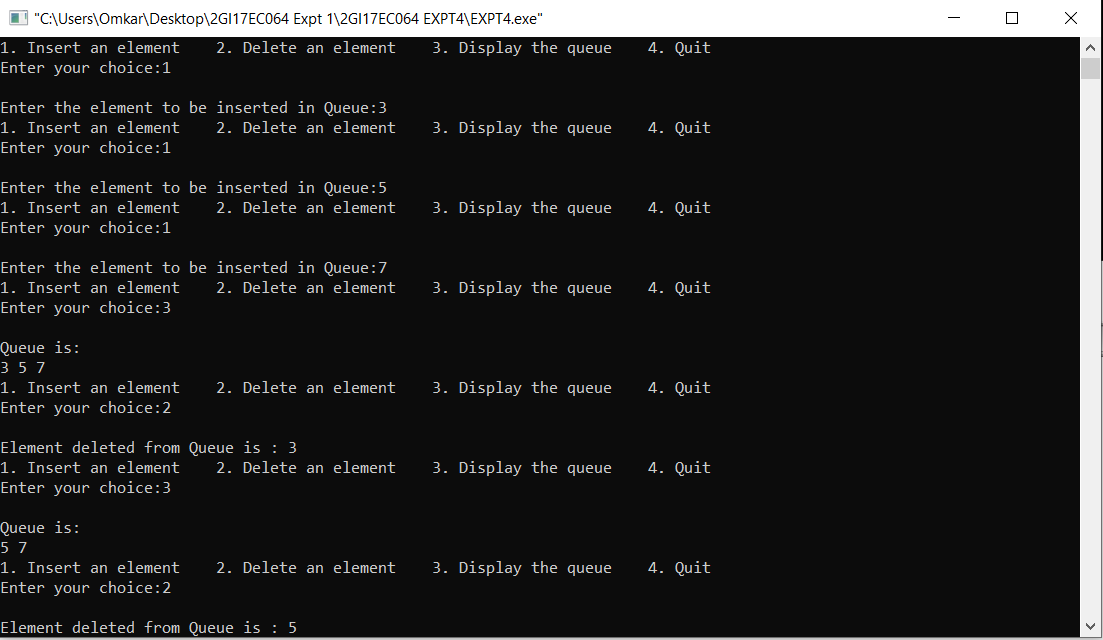
printf("%d ",queue\_arr[i]);

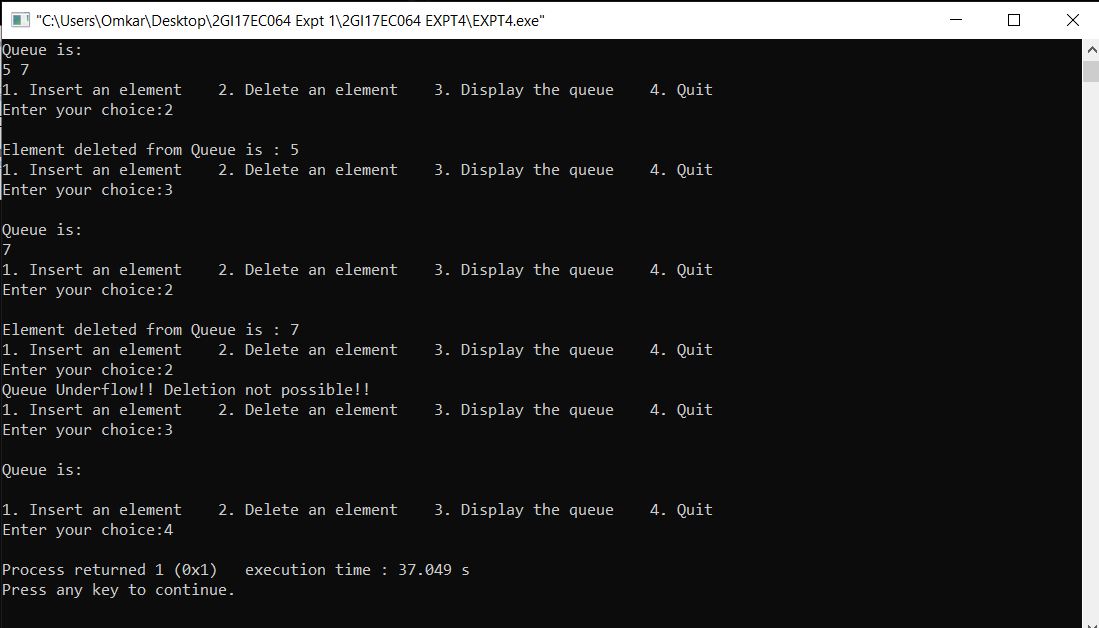
printf("\n");

}

}

**Output:**





**ANALYSIS (LIMITATIONS):**

More memory is required to store elements in linked list as compared to array. Because in linked list each node contains a pointer and it requires extra memory for itself.

Elements or nodes traversal is difficult in linked list.