**Experiment 10**

**Queue Implementation Using Array**

**Date :** 02-10-2020

**Aim:** To implement a Queue using array

**Data Structure used :** Queue, Array

**Algorithms**

**1. Algorithm for enqueue**

Input: An Array implementation of Queue (Q[SIZE]), with front pointing to the first element and rear pointing to the last element in and an element E to be inserted into the queue.

Output: The Queue with the element E inserted at the rear

Data Structure: Queue

Steps:

Step 1: if(rear == SIZE) then

Step 1: print(“The queue is full insertion not possible”)

Step 2: exit(1)

Step 2: else

Step 1: if(rear == -1) then

Step 1: front ++

Step 2: EndIf

Step 3: Q[++rear] = E

Step 3: EndIf

**2. Algorithm for dequeue**

Input: An Array implementation of Queue (Q[SIZE]), with front pointing to the first element and rear pointing to the last element in the queue.

Output: The element E which is removed form the front of the queue

Steps

Step 1: if(front == -1) then

Step 1: print(“The Queue is empty”)

Step 2: exit(1)

Step 2: else

Step 1: E = Q[front]

Step 2: if(front == rear) then

Step 1: front =-1

Step 2: rear =-1

Step 3: else

Step 1: front--

Step 4: endif

Step 3: endif

**Program code:**

/\* Queue implemetation using dynamic array

\* Done By : Rohit Karuankaran

\* \*/

#include <stdlib.h>

#include <stdio.h>

//#define SIZE 50

typedef struct queue\_structure\_datatype

{

int \*Q;

int size;

int front;

int rear;

}queue;

void initQueue(queue \*q)

{

q->size = 16;

q->Q = (int\*) malloc(q->size\*sizeof(int));

q->front = -1;

q->rear = -1;

}

void delQueue(queue \*q)

{

free(q->Q);

}

void incrSize(queue \*q)

{

q->size = 2\*(q->size);

int \*tmp = (int\*) realloc (q->Q,q->size\*sizeof(int));

if(tmp==NULL)

{

printf("Heap is full memory not available");

}

else

{

q->Q = tmp;

}

}

void enQueue(queue \*q,int elem)

{

if(q->rear>=q->size)

{

// printf("The Queue is full Inseriton not possible\n");

incrSize(q);

}

else

{

if(q->front==-1)

{

q->front=q->front+1;

}

q->rear = q->rear+1;

q->Q[q->rear] = elem;

return;

}

}

int deQueue(queue \*q)

{

if(q->front == -1)

{

printf("QUEUE IS EMPTY THERE IS NO ELEMENT TO DELETE\n");

return -1;

}

else

{

int elem = q->Q[q->front];

if(q->front==q->rear)

{

q->front = -1;

q->rear = -1;

}

else

q->front=q->front+1;

return elem;

}

}

void displayQueue(queue \*q)

{

int i = q->front;

if(q->front)

{

printf("EMPTY");

return;

}

while(i>=0&&i<=q->rear)

{

printf("%d ",q->Q[i]);

i++;

}

}

int main()

{

queue \*myQueue = (queue\*) malloc(sizeof(queue));

int RUN = 1;

int elem;

int choice;

initQueue(myQueue);

while(RUN)

{

printf("=======================\n");

printf(" Menu\n");

printf("=======================\n\n");

printf("1.Enter into the queue\n");

printf("2.Remove from the queue\n");

printf("3.Display the queue\n");

printf("4.Exit\n");

printf("Enter your choice : ");

scanf("%d%\*c",&choice);

switch(choice)

{

case 1: printf("Enter the element you want to enter into the Queue : ");

scanf("%d%\*c",&elem);

enQueue(myQueue,elem);

break;

case 2: elem = deQueue(myQueue);

printf("The element remove is :%d\n",elem);

break;

case 3: printf("The Queue is: ");

displayQueue(myQueue);

printf("\n");

break;

case 4: RUN = 0;

break;

default: printf("Enter a valid input\n\n");

}

}

/\*

insert(myQueue,32);

insert(myQueue,21);

displayQueue(myQueue);

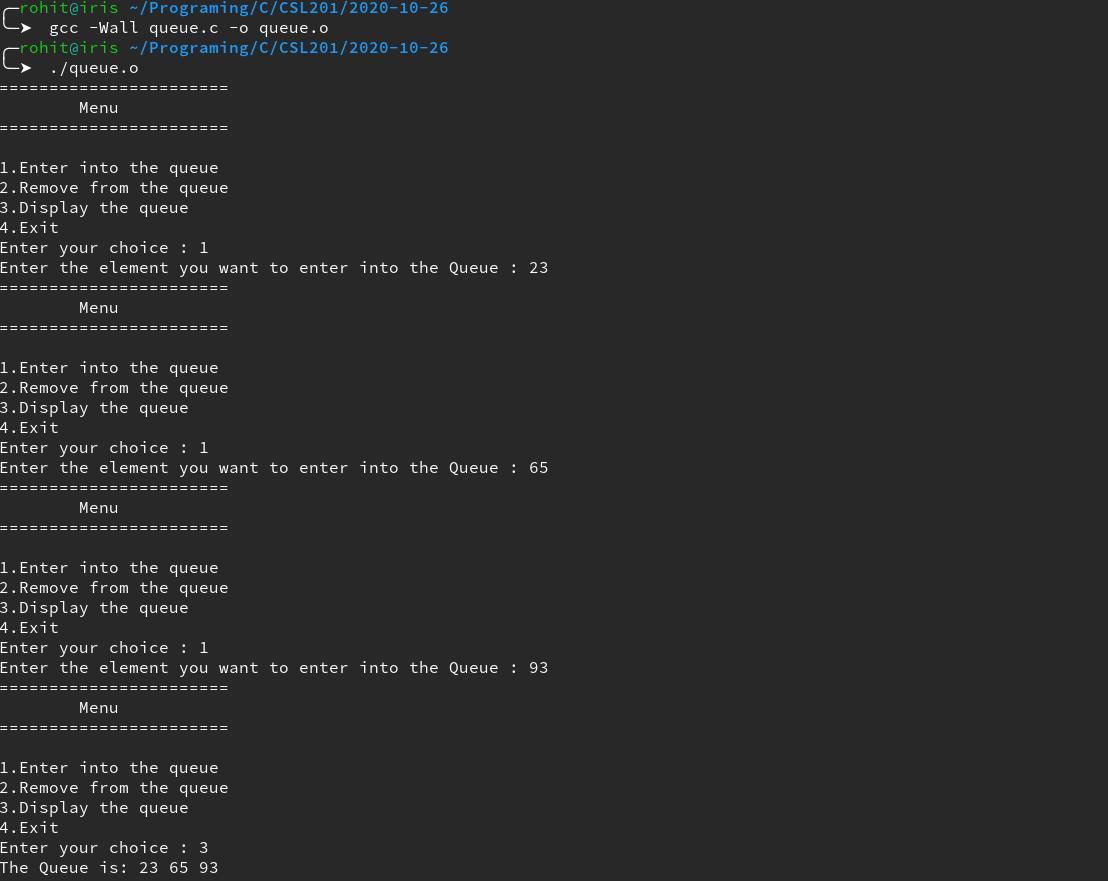
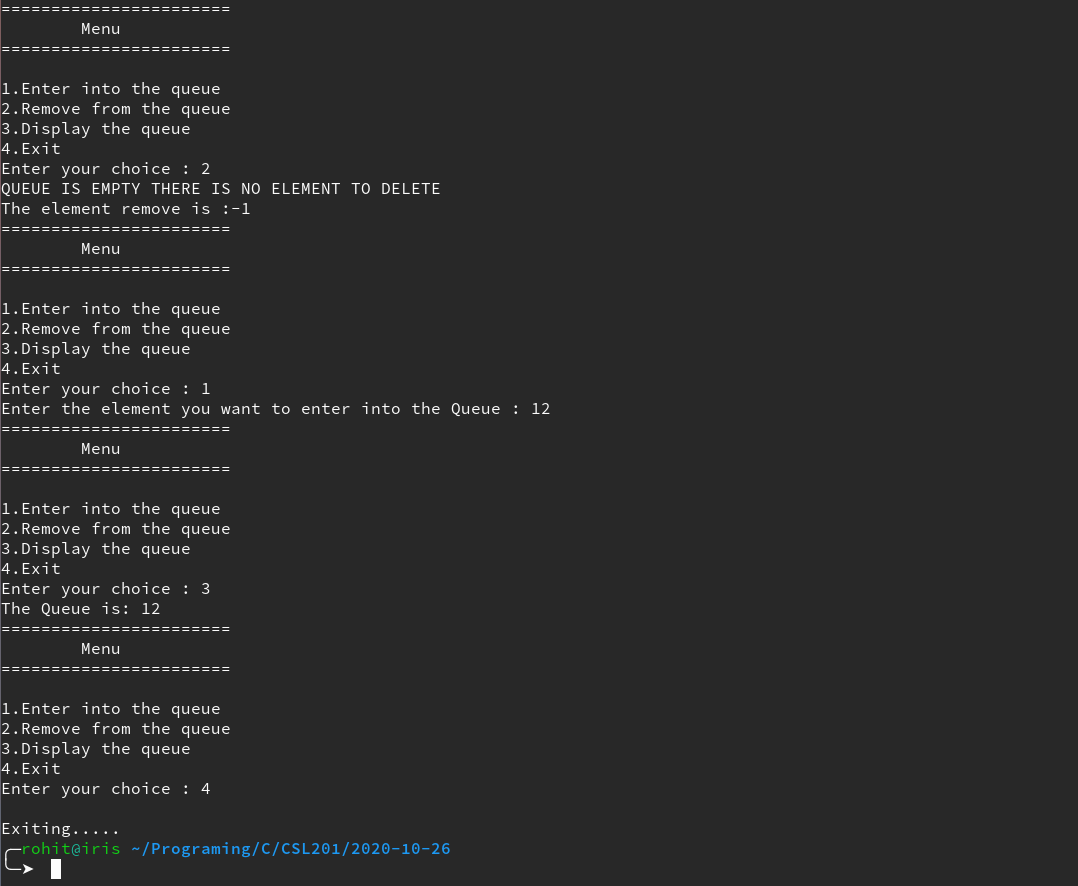
\*/

delQueue(myQueue);

printf("\nExiting.....\n");

}

**Sample input/Output:**

****

**Result:** the Program compiled successfully and the desired output was obtained.