30/10/2020

**Experiment No. 9**

**CIRCULAR QUEUE USING ARRAY**

**AIM:**

Write a program to implement Circular Queue using arrays.

**DATA STRUCTURES USED:**

Queue

**ALGORITHM:**

Algorithm\_ENQUEUE (ITEM)

FRONT=-1 REAR=-1

1. If (FRONT==(REAR+1)%N) // N is the size of Queue
2. print “Queue is full”
3. Exit
4. Else
5. If (FRONT= =-1)
6. FRONT=REAR=0
7. CQueue[REAR]=ITEM
8. Else
9. REAR=(REAR+1)%N
10. CQueue[REAR]=ITEM
11. EndIf
12. EndIf

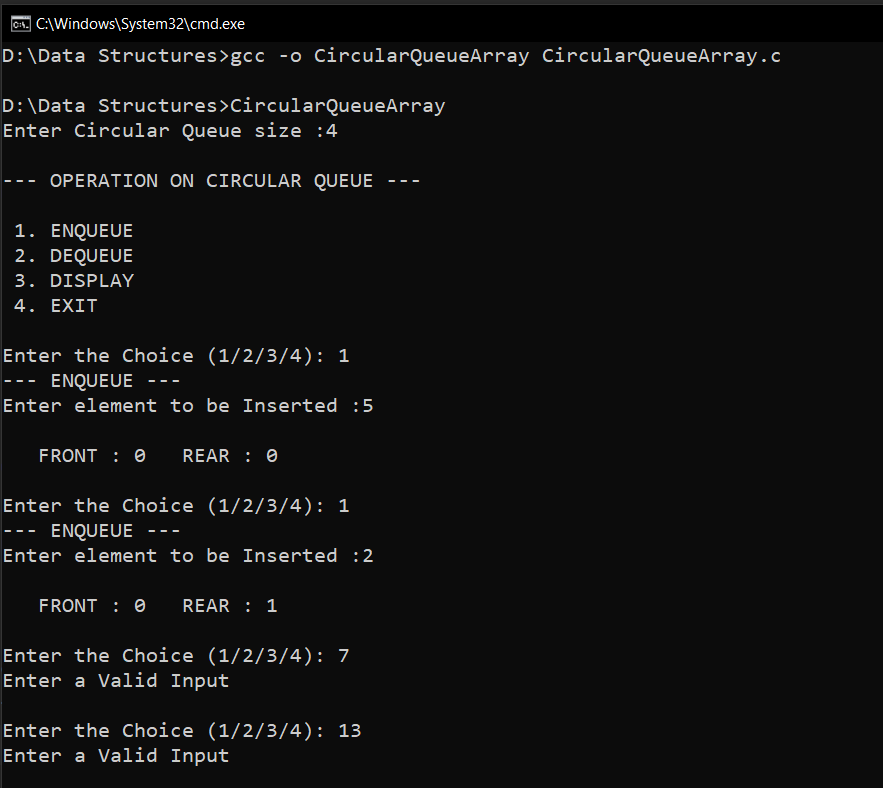
Algorithm\_DEQUEUE ()

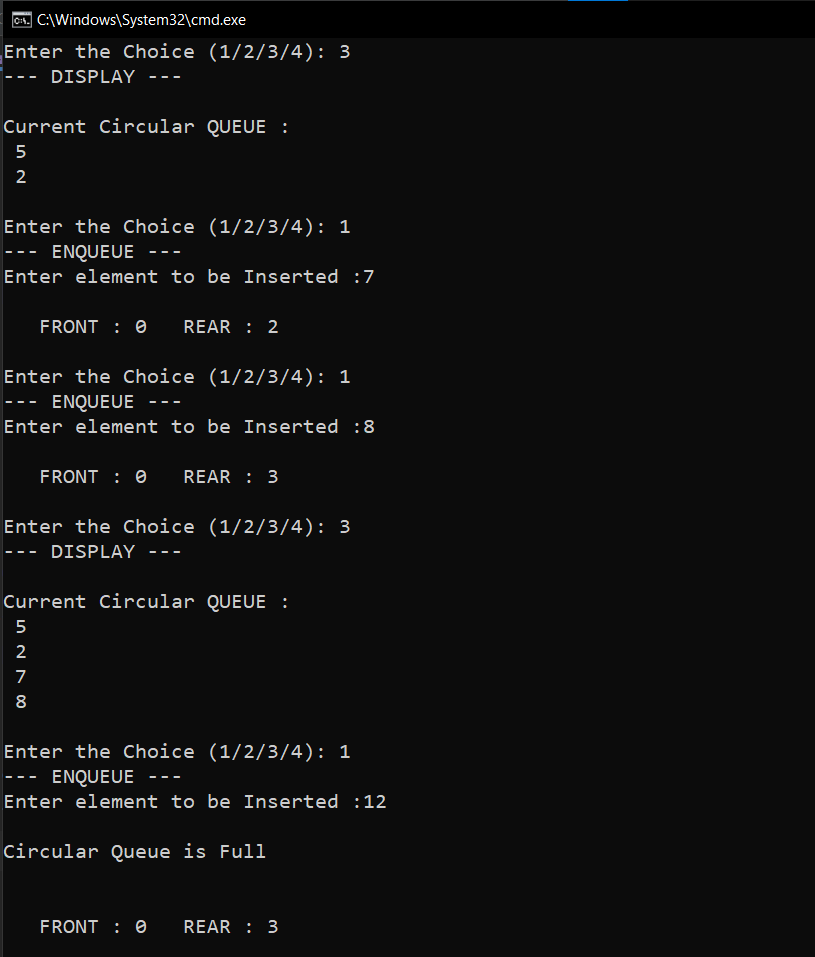
1. If (FRONT=-1)
2. Print “Queue is empty”
3. Exit
4. Else
5. If (FRONT == REAR)
6. ITEM = CQueue[FRONT]
7. FRONT=REAR=-1
8. Else
9. FRONT=(FRONT+1)%N
10. ITEM = CQueue[FRONT]
11. EndIf
12. EndIf

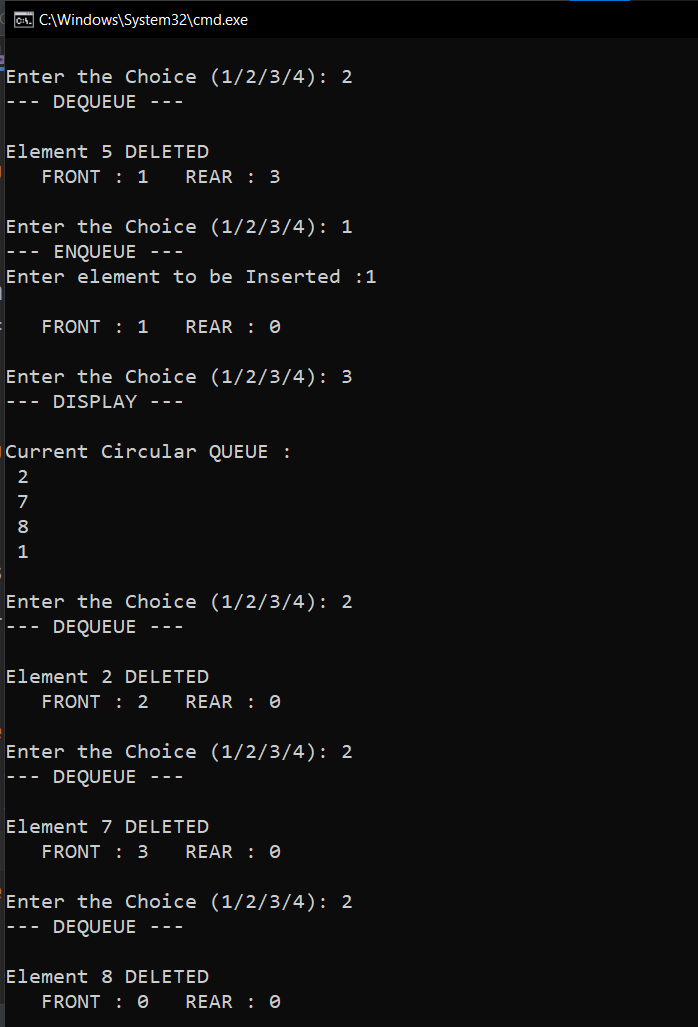
**PROGRAM:**

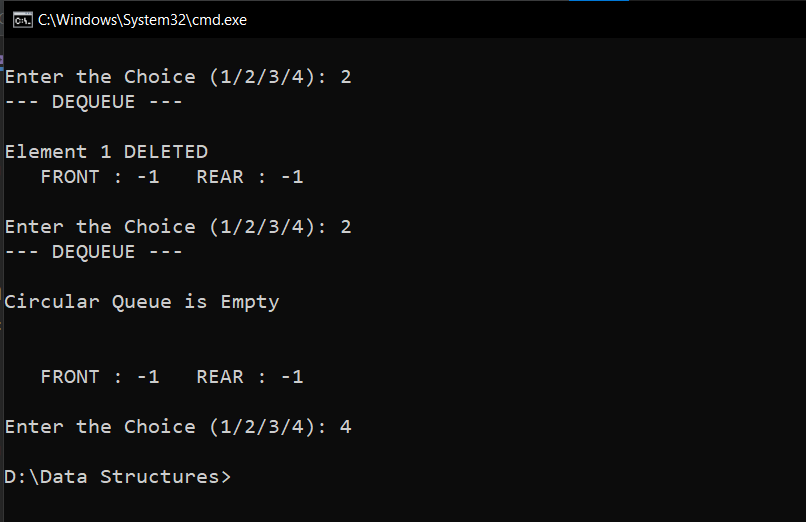
#include <stdio.h>  
#include <stdlib.h>  
int size;  
int front;  
int rear;  
int \*arr;  
  
int isFull(){  
 if(front==(rear+1)%size){  
 return 1;  
 }  
 return 0;  
}  
  
int isEmpty(){  
 if(front==-1){  
 return 1;  
 }  
 return 0;  
}  
  
void insert(int item){  
 if(isFull()){  
 printf("\nCircular Queue is Full\n\n");  
 }  
 else if(front==-1){  
 arr[++rear] = item;  
 front++;  
 }  
 else {  
 rear=(rear+1)%size;  
 arr[rear] = item;  
 }  
 printf("\n FRONT : %d REAR : %d \n",front,rear);  
}  
void delete(){  
 if(isEmpty()){  
 printf("\nCircular Queue is Empty\n\n");  
 }  
 else if(front==rear){  
 int item = arr[front];  
 printf("\nElement %d DELETED ",item);  
 front=-1;  
 rear=-1;  
  
 }else{  
 int item = arr[front];  
 front=(front+1)%size;  
 printf("\nElement %d DELETED ",item);  
 }  
 printf("\n FRONT : %d REAR : %d \n",front,rear);  
}  
  
void display(){  
 printf("\nCurrent Circular QUEUE :\n");  
 if(isEmpty()){  
 printf("\nCircular Queue is Empty \n");  
 }else if (rear >= front){  
 for(int i=front;i<=rear;i++){  
 printf(" %d\n",arr[i]);  
 }  
 }  
 else{  
 for(int i=front;i<size;i++){  
 printf(" %d\n",arr[i]);  
 }  
 for(int i=0;i<=rear;i++){  
 printf(" %d\n",arr[i]);  
 }  
 }  
}  
  
void main(){  
 int n,x,y;  
 char ans='y';  
 printf("Enter Circular Queue size :");  
 scanf("%d", &size);  
 arr = (int\*) malloc (size \* sizeof(int));  
 front=-1,rear=-1;  
 printf("\n--- OPERATION ON CIRCULAR QUEUE --- \n\n");  
 printf(" 1. ENQUEUE \n");  
 printf(" 2. DEQUEUE\n");  
 printf(" 3. DISPLAY\n");  
 printf(" 4. EXIT\n");  
 while(ans=='y'){  
 printf("\nEnter the Choice (1/2/3/4): ");  
 scanf("%d",&n);  
 switch(n){  
 case 1:printf("--- ENQUEUE ---\n");  
 printf("Enter element to be Inserted :");  
 scanf("%d", &x);  
 insert(x);  
 break;  
 case 2:printf("--- DEQUEUE ---\n");  
 delete();  
 break;  
 case 3:printf("--- DISPLAY ---\n");  
 display();  
 break;  
 case 4:ans='n';  
 break;  
 default:printf("Enter a Valid Input\n");  
 }  
 }  
}

**OUTPUT:**









**RESULT:**

The Enqueue() and Dequeue() operation on circular queue was implemented.

Time complexity of Enqueue() operation is O(1).

Time complexity of Dequeue() operation is O(1).