Experiment No.5
Implement Circular Queue ADT using array
Name:Tejashree Anand Karekar
Roll No: 20
Date of Performance:
Date of Submission:
Marks:
Sign:

Experiment No. 5: Circular Queue

Aim: To Implement Circular Queue ADT using array

Objective:

Circular Queues offer a quick and clean way to store FIFO data with a maximum size

Theory:

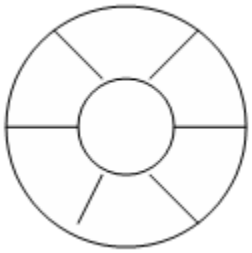
Circular queue is an data structure in which insertion and deletion occurs at an two ends rear and front respectively. Eliminating the disadvantage of linear queue that even though there is a vacant slots in array it throws full queue exception when rear reaches last element. Here in an circular queue if the array has space it never throws an full queue exception. This feature needs an extra variable count to keep track of the number of insertion and deletion in the queue to check whether the queue is full or not.Hence circular queue has better space utilization as compared to linear queue. Figure below shows the representation of linear and circular queue.

Linear queue

Front rear

0	…	…		n

Circular Queue



Algorithm

Algorithm : ENQUEUE(Item)

Input : An item is an element to be inserted in a circular queue.

Output : Circular queue with an item inserted in it if the queue has an empty slot.

Data Structure : Q be an array representation of a circular queue with front and rear pointing to the first and last element respectively.

1. If front = 0
   front = 1
   rear =1
   Q[front] = item
2. else
   next=(rear mod length)
   if next!=front then
   rear = next
   Q[rear] = item
   Else
   Print "Queue is full"
   End if

End if

3. stop

Algorithm : DEQUEUE()

Input : A circular queue with elements.

Output :Deleted element saved in Item.

Data Structure : Q be an array representation of a circular queue with front and rear pointing to the first and last element respectively.

1. If front = 0
   Print "Queue is empty"
   Exit
2. else
   item = Q[front]
   if front = rear then
   rear = 0
   front=0
   else
   front = front+1
   end if
   end if
3. stop

Code:

```
#include <stdio.h>

#include <conio.h>

#define MAX 10
```

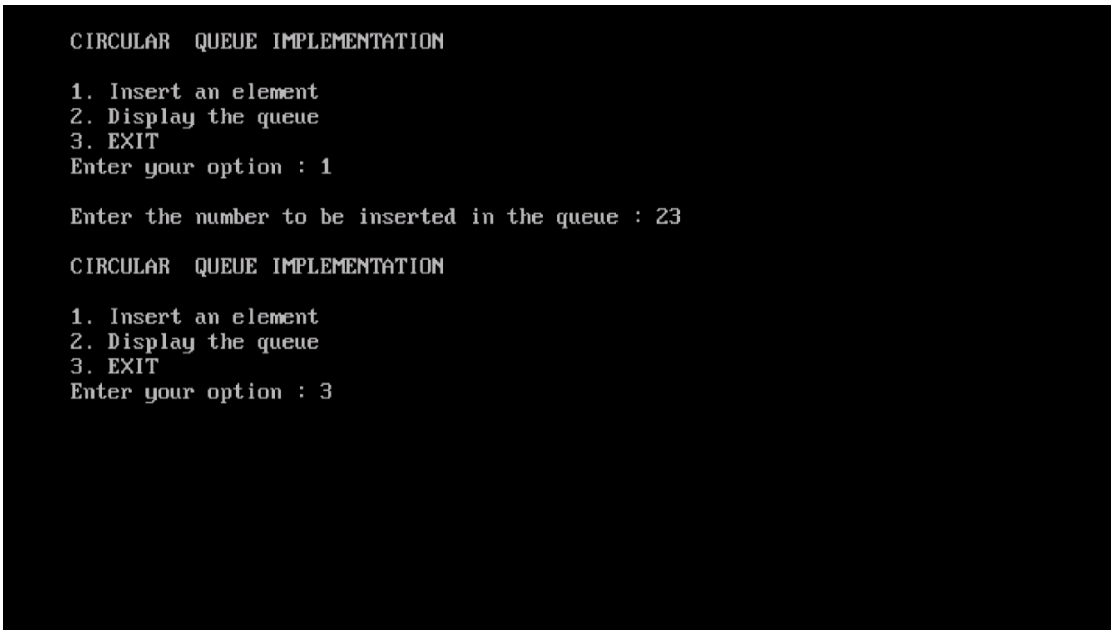
```
int queue[MAX];

int front=-1, rear=-1;

void insert(void);


void display(void);

int main()

{

int option;

clrscr();

do

{

 printf("\n CIRCULAR  QUEUE IMPLEMENTATION ");

 printf("\n");

 printf("\n 1. Insert an element");

 printf("\n 2. Display the queue");

 printf("\n 3. EXIT");

 printf("\n Enter your option : ");

 scanf("%d", &option);

switch(option)

 {

 case 1:

 insert();

break;

case 2:
```

```
 display();
break;
 }
}while(option!=3);
getch();
return 0;
}
void insert()
{
int num;
printf("\n Enter the number to be inserted in the queue : ");
scanf("%d", &num);
if(front==0 && rear==MAX-1)
 printf("\n OVERFLOW");
else if(front==-1 && rear==-1)
{
front=rear=0;
 queue[rear]=num;
}
else if(rear==MAX-1 && front!=0)
{
rear=0;
 queue[rear]=num;
}
```

```
else
{
 rear++;
 queue[rear]=num;
}
}
void display()
{
int i;
printf("\n");
if (front ==-1 && rear==-1)
 printf ("\n QUEUE IS EMPTY");
else
{
 if(front<rear)
 {
 for(i=front;i<=rear;i++)
 printf("\t %d", queue[i]);
 }
 else
 {
 for(i=front;i<MAX;i++)
 printf("\t %d", queue[i]);
for(i=0;i<=rear;i++)
```

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

Output:



Conclusion:

1)Explain how Josephus Problem is resolved using circular queue and elaborate on operation used for the same.

The Josephus Problem is a theoretical problem that involves a group of people standing in a circle, and they are eliminated one by one in a fixed pattern until only one person remains. To solve this problem using a circular queue, you can follow these steps:

1. Create a circular queue and enqueue all the people in the circle, assigning them sequential numbers (e.g., 1 to N).

2. Define the elimination step, which determines how many positions to move around the circle before eliminating a person. Let's say you want to eliminate every Mth person.

3. Start at the front of the circular queue.

4. Dequeue the first person and count one position.

5. If the count reaches M, eliminate that person by not enqueueing them back into the queue.

6. Continue this process, dequeuing and counting, until only one person remains.

7. The last person remaining is the solution to the Josephus Problem.

Here are the main operations used in solving the Josephus Problem using a circular queue:

1. Enqueue: Add a person to the back of the circular queue.

2. Dequeue: Remove and return the person at the front of the circular queue.

3. Counting: Keep track of positions as you move around the circle, and when the count reaches the predetermined value (M), eliminate the person at that position.

4. Circular Movement: Ensure that when you reach the end of the queue, you wrap around to the front to create a circular structure.

By using these operations in a circular queue, you can efficiently simulate the elimination process and find the last person remaining in the Josephus Problem.