**Program 5**

**Aim :** To implement a circular queue

**Theory :** Circular Queue is a linear data structure in which the operations are performed based on FIFO (First In First Out) principle and the last position is connected back to the first position to make a circle. It is also called **‘Ring Buffer’**.

Circular Queue works by the process of circular increment i.e. when we try to increment any variable and we reach the end of queue, we start from the beginning of queue by modulo division with the queue size. In a normal Queue, we can insert elements until queue becomes full. But once queue becomes full, we cannot insert the next element even if there is a space in front of queue. This drawback is overcome by circular queue.

**Algorithm :**

Enqueue(queue\*q)

1. If q->size == MAXSIZE then
2. Print(“Overflow”)
3. Terminate function.
4. Take input data from user
5. If q->size == 0 then
6. Set q->front = 0 and q->rear = 0
7. Else
8. Set q->rear = (q->rear + 1)%MAXSIZE
9. Set q->arr[q->rear] = data
10. Increment q->size by 1

Dequeue(queue\*q)

1. If q->size == 0 then
2. Print(“Underflow”)
3. Terminate the function.
4. Set data = q->arr[q->front]
5. Set q->front = (q->front + 1)%MAXSIZE
6. Decrease q->size by 1
7. Print(“data removed from queue”)

Display(queue \* q)

1. If q->size == 0 then
2. Print(“Queue is empty!!!”)
3. Terminate the function.
4. Set temp = q->front
5. Set i = 0
6. While i<q->size repeat steps I to III
7. Print(q->arr[temp])
8. Increment i
9. Set temp = (temp+1)%MAXSIZE

**Program :**

#include <stdio.h>

#include <stdlib.h>

#define MAXSIZE 10

struct queue{

int arr[MAXSIZE];

int front;

int rear;

int size;

};

void enqueue(struct queue\*);

void dequeue(struct queue\*);

void display(struct queue\* );

int main(){

int choice;

struct queue q1;

q1.size = 0;

do{

printf("\n1. Enqueue\n");

printf("2. Dequeue\n");

printf("3. Display\n");

printf("4. Exit\n");

printf("Enter your choice : ");

scanf("%d",&choice);

switch(choice){

case 1:

enqueue(&q1);

break;

case 2:

dequeue(&q1);

break;

case 3:

display(&q1);

break;

default:

break;

}

}while(choice<4&&choice>0);

return 0;

}

void enqueue(struct queue \* q){

if(q->size == MAXSIZE){

printf("\nOverflow\n");

return;

}

int data;

printf("Enter value to be inserted : ");

scanf("%d",&data);

if(q->size == 0){

q->front = q->rear = 0;

}

else

q->rear = (q->rear + 1)%MAXSIZE;

q->arr[q->rear] = data;

q->size++;

}

void dequeue(struct queue \* q){

if(q->size == 0){

printf("\nUnderFlow\n");

return;

}

int data = q->arr[q->front];

q->front = (q->front + 1)%MAXSIZE;

q->size--;

printf("\n%d removed from queue\n",data);

}

void display(struct queue \* q){

printf("\n");

if(!q->size){

printf("Empty Queue!!!\n");

return;

}

int temp = q->front;

int i = 0;

for( ; i<q->size ; i++){

printf("%d ",q->arr[temp]);

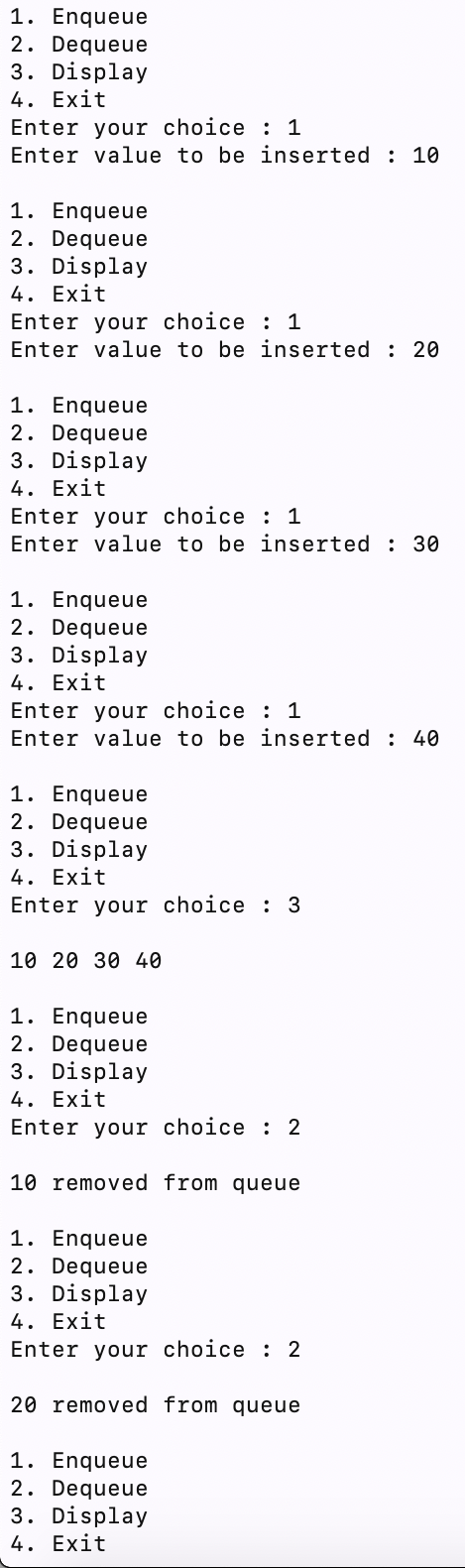
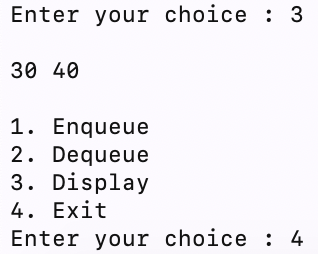
temp = (temp + 1)%MAXSIZE;

}

printf("\n");

}

**OUTPUT :**

**Learning :** We learnt how to implement a circular queue using an array.