Yash Sarang.
Roll No: 47, Class : D6AD.
Data Structures. Experiment-05.

AIM: Implement Linear Queue ADT using array.

Theory: A queue is a useful data structure in programming. It is similar to the ticket queue outside a cinema hall, where the first person entering the queue is the first person who gets the ticket.

Queue follows the First In First Out (FIFO) rule - the item that goes in first is the item that comes out first.
• Queue is a linear data structure which follows First In First Out (FIFO) principle
• Queue is a linear list of elements of same type in which insertion of an element is performed at one end and deletion
• of an element is performed at another end • Insertion can take place only at one end which is called as Rear
• Deletion can take place only at one end which is called as Front
Basic Operations of Queue.

A queue is an object (an abstract data structure - ADT) that allows the following operations:
1.Enqueue: Add an element to the end of the queue
2.Dequeue: Remove an element from the front of the queue
3.IsEmpty: Check if the queue is empty
4.IsFull: Check if the queue is full
5.Peek: Get the value of the front of the queue without removing it .

Algorithm to insert any element in a queue:

Check if the queue is already full by comparing rear to max - 1. if so, then return an overflow error.

If the item is to be inserted as the first element in the list, in that case set the value of front and rear to 0 and insert the element at the rear end. Otherwise keep increasing the value of rear and insert each element one by one having rear as the index.

Algorithm

```
Step 1:     IF REAR = MAX - 1
                Write OVERFLOW
                Go to step 4
            [END OF IF]

Step 2:     IF FRONT = -1 and REAR = -1
                SET FRONT = REAR = 0
            ELSE
                SET REAR = REAR + 1
            [END OF IF]

Step 3:     Set QUEUE[REAR] = NUM

Step 4:     EXIT
```

Algorithm to delete an element from the queue
If the value of front is -1 or value of front is greater than rear , write an underflow message and exit.

Otherwise, keep increasing the value of front and return the item stored at the front end of the queue at each time

Algorithm

```
Step 1:     IF FRONT = -1 or FRONT > REAR
                Write UNDERFLOW
            ELSE
                SET VAL = QUEUE[FRONT]
                SET FRONT = FRONT + 1
            [END OF IF]

Step 2:     EXIT
```

C Program:

```
#include <stdio.h>
#define SIZE 10

void enque(int);
void deQuene();
void display();
int array[SIZE], front = -1, rear = -1;

void main()
{
	int choice, a;
	do
	{
		printf("\n ***** Circular Queue ****");
		printf("\n 1. Insert an Element");
		printf("\n 2. Delete an Element");
		printf("\n 3. Display The Queue");
		printf("\n Enter a choice");
		scanf("%d", &choice);
		switch (choice)
		{
			case 1:
				printf("\n Enter the element to be inserted : ");
				scanf("%d", &a);
				enque(a);
				break;
			case 2:
				deQuene();
				break;
			case 3:
				display();
				break;
			default:
				printf("Invalid Input");
				break;
		}
	}while (choice <4 );
}
void enque(int value)
```

```
{
        if (rear == SIZE - 1)
        {
                printf("\n Queue is Full");
        }
        else
        {
                if (front == -1)
                        front = 0;
                 rear++;
                 array[rear] = value;
                 printf("\n Inserted item is %d", value);
         }
}
void deQueue()
{
        if (front == -1)
        {
                printf("\n Queue is empty");
        }
        else
        {
                printf("\n deleted : %d", array[front]);
                front++;
                if (front > rear)
                front = rear = -1;
        }
}
void display()
{
        if (rear == -1)
        {
                printf("\n Queue is Empty! \n");
        }
        else
        {
                printf("\n Elements in Queue Re: ");
                for (int i = front; i <= rear; i++)
                {
                        printf("%d ", array[i]);
```

```
        }
    }
}
```

OUTPUT:

```
***** Circular Quene ****
1. Insert an Element
2. Delete an Element
3. Dispaly The Quene
Enter a choice1

Enter the element to be inserted : 34

insereted item is 34
***** Circular Quene ****
1. Insert an Element
2. Delete an Element
3. Dispaly The Quene
Enter a choice1

Enter the element to be inserted : 45

insereted item is 45
***** Circular Quene ****
1. Insert an Element
2. Delete an Element
3. Dispaly The Quene
Enter a choice3

Elements in Quene Re: 34 45
***** Circular Quene ****
1. Insert an Element
2. Delete an Element
3. Dispaly The Quene
Enter a choice2

deleted : 34
***** Circular Quene ****
1. Insert an Element
2. Delete an Element
3. Dispaly The Quene
```
