NAME:- SAGAR ADHIKARI
ROLL NO:- PUR078BEI034
FACULTY:- ELECTRONICS

DSA LAB SHEET NO. 2

TITLE:- STATIC IMPLEMENTATION OF QUEUE USING ARRAY

THEORY:-

Queue:-A Queue is defined as a linear data structure that is open at both ends and the operations are performed in First In First Out (FIFO) order.
Array:- An array is a collection of items of same data type stored at contiguous memory locations.

PROGRAM CODE:-

```
#include <string.h>
#include <stdio.h>
#define TRUE 1
#define FALSE 0
#define MAX 5
#define strlen 20
typedef struct QUEUE
{
	int FRONT;
	int REAR;
	char DATA[MAX][strlen];
} Queuest;			// Renaming so have not to write (struct QUEUE) everytime
int IsFull(Queuest *s);	// Function prototyping
int IsEmpty(Queuest *s);
void Enqueue(Queuest *s, char *enteredstring);
char *Dequeue(Queuest *s);
int main()
{
	int choice;
	char value[strlen];
	Queuest Queue = {0, -1};
	do
	{
		printf("\n1.Enqueue\n2.Dequeue\n3.Exit\n");
		scanf("%d", &choice);
		switch (choice)
		{
		case 1:
```

DSA LAB SHEET NO. 2

```
                if (IsFull(&Queue))
                        printf("Queue is full\n");
                else
                {
                        printf("enter string element:");
                        scanf("%s", &value);
                        Enqueue(&Queue, value);
                        printf("%s was Enqueued\n", value);
                }

                break;
        case 2:
                if (IsEmpty(&Queue))
                        printf("Queue is empty\n");
                else
                        printf("%s was    Dequeued\n", Dequeue(&Queue));
                break;
        case 3:
                printf("Exited from the program\n");
                break;

        default:
                printf("enter 1,2, or 3 only\n");
                break;
        }
    } while (choice != 3);
    printf("\n");
    return 0;
}
int IsFull(Queuest *s)
{
    return (s->REAR == MAX - 1) ? TRUE : FALSE;
}
int IsEmpty(Queuest *s)
{
    return s->REAR < s->FRONT ? TRUE : FALSE;
}
void Enqueue(Queuest *s, char *enteredstring)
{
    strcpy(s->DATA[++s->REAR], enteredstring);
}
```

NAME:- SAGAR ADHIKARI
ROLL NO:- PUR078BEI034
FACULTY:- ELECTRONICS

DSA LAB SHEET NO. 2

```
char *Dequeue(Queuest *s)       // returning an array requires pointer.
{
        return s->DATA[s->FRONT++];
}
```

OUTPUT:-

```
1.Enqueue
2.Dequeue
3.Exit
1
enter string element:Dell
Dell was Enqueued

1.Enqueue
2.Dequeue
3.Exit
1
enter string element:Acer
Acer was Enqueued

1.Enqueue
2.Dequeue
3.Exit
1
enter string element:Asus
Asus was Enqueued

1.Enqueue
2.Dequeue
3.Exit
1
enter string element:Apple
Apple was Enqueued

1.Enqueue
2.Dequeue
3.Exit
1
enter string element:Lenovo
```

NAME:- SAGAR ADHIKARI
ROLL NO:- PUR078BEI034
FACULTY:- ELECTRONICS

DSA LAB SHEET NO. 2

Lenovo was Enqueued

1.Enqueue
2.Dequeue
3.Exit
1
Queue is full

1.Enqueue
2.Dequeue
3.Exit
2
Dell was Dequeued

1.Enqueue
2.Dequeue
3.Exit
2
Acer was Dequeued

1.Enqueue
2.Dequeue
3.Exit
2
Asus was Dequeued

1.Enqueue
2.Dequeue
3.Exit
2
Apple was Dequeued

1.Enqueue
2.Dequeue
3.Exit
2
Lenovo was Dequeued

1.Enqueue
2.Dequeue
3.Exit

NAME:- SAGAR ADHIKARI
ROLL NO:- PUR078BEI034
FACULTY:- ELECTRONICS

DSA LAB SHEET NO. 2

2
Queue is empty

1.Enqueue
2.Dequeue
3.Exit
4
enter 1,2, or 3 only

1.Enqueue
2.Dequeue
3.Exit
3
Exited from the program