**Assignment: 9**

**EXP.** WAP to implement the following scenarios.

1. Create Queue
2. Perform Enqueue and Deque operations on Queue.
3. Traverse the queue and print its element.
4. Print underflow and overflow when desired conditions are not met.
5. Reverse the elements of Queue using recursion.

**CODING:**

#include <stdio.h>

# define SIZE 20

void enqueue();

void dequeue();

void show();

int inp\_arr[SIZE];

int Rear = - 1;

int Front = - 1;

main()

{

    int a;

    while (1)

    {

        printf("1.Enqueue Operation\n");

        printf("2.Dequeue Operation\n");

        printf("3.Display the Queue\n");

        printf("4.Exit\n");

        printf("Enter your choice of operations : ");

        scanf("%d", &a);

        switch (a)

        {

            case 1:

            enqueue();

            break;

            case 2:

            dequeue();

            break;

            case 3:

            show();

            break;

            case 4:

            exit(0);

            default:

            printf("Incorrect choice \n");

        }

    }

}

void enqueue()

{

    int insert\_item;

    if (Rear == SIZE - 1)

       printf("Overflow \n");

    else

    {

        if (Front == - 1)

        Front = 0;

        printf("Element to be inserted in the Queue\n : ");

        scanf("%d", &insert\_item);

        Rear = Rear + 1;

        inp\_arr[Rear] = insert\_item;

    }

}

void dequeue()

{

    if (Front == - 1 || Front > Rear)

    {

        printf("Underflow \n");

        return ;

    }

    else

    {

        printf("Element deleted from the Queue: %d\n", inp\_arr[Front]);

        Front = Front + 1;

    }

}

void show()

{

    if (Front == - 1)

        printf("Empty Queue \n");

    else

    {

        printf("Queue: \n");

        for (int i = Front; i <= Rear; i++)

            printf("%d ", inp\_arr[i]);

        printf("\n");

    }

}

**OUTPUT:**



