**ASSIGNMENT 6:**

**1.Write a program for implementing insert,delete and display**

**operation of circular queue.**

**Sol:**

#include <stdio.h>

# define max 6

int queue[max];  // array declaration

int front=-1;

int rear=-1;

void enqueue(int element)

{

    if(front==-1 && rear==-1)

    {

        front=0;

        rear=0;

        queue[rear]=element;

    }

    else if((rear+1)%max==front)      {

        printf("Queue is overflow..");

    }

    else

    {

        rear=(rear+1)%max;

        queue[rear]=element;         }

}

int dequeue()

{

    if((front==-1) && (rear==-1))

    {

        printf("\nQueue is underflow..");

    }

 else if(front==rear)

{

   printf("\nThe dequeued element is %d", queue[front]);

   front=-1;

   rear=-1;

}

else

{

    printf("\nThe dequeued element is %d", queue[front]);

   front=(front+1)%max;

}

}

void display()

{

    int i=front;

    if(front==-1 && rear==-1)

    {

        printf("\n Queue is empty..");

    }

    else

    {

        printf("\nElements in a Queue are :");

        while(i<=rear)

        {

            printf("%d,", queue[i]);

            i=(i+1)%max;

        }

    }

}

int main()

{

    int choice=1,x;   // variables declaration

    while(choice<4 && choice!=0)   // while loop

    {

    printf("\n Press 1: Insert an element");

    printf("\nPress 2: Delete an element");

    printf("\nPress 3: Display the element");

    printf("\nEnter your choice");

    scanf("%d", &choice);

    switch(choice)

    {

        case 1:

        printf("Enter the element which is to be inserted");

        scanf("%d", &x);

        enqueue(x);

        break;

        case 2:

        dequeue();

        break;

        case 3:

        display();

    }}

    return 0;

}

**2.** **A Barua number is a number which consists of only zeroes and ones and has only one 1. Barua number will start with 1. Given numbers, find out the multiplication of the numbers. Note: The input may contain one decimal number and all other Barua numbers. (Assume that each number is very large and total number of values give is also very large)**

**Input 1: 100 10 12 1000**

**Output 1: 12000000**

**Input 2: 100 121 1000000000000000**

**Output 2: 12100000000000000000**

**Input 3: 10 100 1000**

**Output 3: 1000000**

**Sol:**

* The only thing that we have to count the number of zeros in each of the numbers and simultaneously delete the zero in that number.
* So now we have numbers with no trailing zeros and a count variable with the number of zeros in every number.
* The counting is done by using divide(/) and modulus(%) with 10 and also using loops(mainly while loop is used).
* Finally multiple the numbers with no trailing zeros and using a loop print zeros(as such as the number in count variable) at the end.

**3.** **Implement push, pop and find the minimum element in a stack in O(1) time complexity.**

**Sol:**

public class Stack<T> {

private class Item {

public T Value;

public Item Next;

}

private Item First;

private Stack MinStack;

public Stack<T>() {

Stack = new Stack<T>();

}

public T Peek() {

return First.Value;

}

public void Push(T value) {

Item item = new Item();

item.Value = value;

item.Next = First;

if (Value < MinStack.Peek()) {

MinStack.Push(Value);

} else {

MinStack.Push(MinStack.Peek());

}

First = item;

}

public T Pop() {

Item item = First;

First = First.Next;

MinStack.Pop();

return item.Value;

}

public T Min() {

return MinStack.Peek();

}