**Question 1**

**Write a program implementing insert, delete and display operation of Circular Queue.**

**Ans:**

**#include<stdio.h>**

**# define MAX 5**

**int cqueue\_arr[MAX];**

**int front = -1;**

**int rear = -1;**

**void insert(int item)**

**{**

**if((front == 0 && rear == MAX-1) || (front == rear+1))**

**{**

**printf("Queue Overflow n");**

**return;**

**}**

**if(front == -1)**

**{**

**front = 0;**

**rear = 0;**

**}**

**else**

**{**

**if(rear == MAX-1)**

**rear = 0;**

**else**

**rear = rear+1;**

**}**

**cqueue\_arr[rear] = item ;**

**}**

**void deletion()**

**{**

**if(front == -1)**

**{**

**printf("Queue Underflown");**

**return ;**

**}**

**printf("Element deleted from queue is : %dn",cqueue\_arr[front]);**

**if(front == rear)**

**{**

**front = -1;**

**rear=-1;**

**}**

**else**

**{**

**if(front == MAX-1)**

**front = 0;**

**else**

**front = front+1;**

**}**

**}**

**void display()**

**{**

**int front\_pos = front,rear\_pos = rear;**

**if(front == -1)**

**{**

**printf("Queue is emptyn");**

**return;**

**}**

**printf("Queue elements :n");**

**if( front\_pos <= rear\_pos )**

**while(front\_pos <= rear\_pos)**

**{**

**printf("%d ",cqueue\_arr[front\_pos]);**

**front\_pos++;**

**}**

**else**

**{**

**while(front\_pos <= MAX-1)**

**{**

**printf("%d ",cqueue\_arr[front\_pos])**

**front\_pos++;**

**}**

**front\_pos = 0;**

**while(front\_pos <= rear\_pos)**

**{**

**printf("%d ",cqueue\_arr[front\_pos]);**

**front\_pos++;**

**}**

**}**

**printf("n");**

**}**

**int main()**

**{**

**int choice,item;**

**do**

**{**

**printf("1.Insertn");**

**printf("2.Deleten");**

**printf("3.Displayn");**

**printf("4.Quitn");**

**printf("Enter your choice : ");**

**scanf("%d",&choice);**

**switch(choice)**

**{**

**case 1 :**

**printf("Input the element for insertion in queue : ");**

**scanf("%d", &item);**

**insert(item);**

**break;**

**case 2 :**

**deletion();**

**break;**

**case 3:**

**display();**

**break;**

**case 4:**

**break;**

**default:**

**printf("Wrong choicen");**

**}**

**}while(choice!=4);**

**return 0;**

**}**

**Question 2**

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

**Ans:**

**struct MyStack**

**{**

**stack<int> s;**

**int minEle;**

**void getMin()**

**{**

**if (s.empty())**

**cout << "Stack is empty\n";**

**else**

**cout <<"Minimum Element in the stack is: "**

**<< minEle << "\n";**

**}**

**void peek()**

**{**

**if (s.empty())**

**{**

**cout << "Stack is empty ";**

**return;**

**}**

**int t = s.top(); // Top element.**

**cout << "Top Most Element is: ";**

**(t < minEle)? cout << minEle: cout << t;**

**}**

**void pop()**

**{**

**if (s.empty())**

**{**

**cout << "Stack is empty\n";**

**return;**

**}**

**cout << "Top Most Element Removed: ";**

**int t = s.top();**

**s.pop();**

**if (t < minEle)**

**{**

**cout << minEle << "\n";**

**minEle = 2\*minEle - t;**

**}**

**else**

**cout << t << "\n";**

**}**

**void push(int x)**

**{**

**if (s.empty())**

**{**

**minEle = x;**

**s.push(x);**

**cout << "Number Inserted: " << x << "\n";**

**return;**

**}**

**// If new number is less than minEle**

**if (x < minEle)**

**{**

**s.push(2\*x - minEle);**

**minEle = x;**

**}**

**else**

**s.push(x);**

**cout << "Number Inserted: " << x << "\n";**

**}**

**};**

**// Driver Code**

**int main()**

**{**

**MyStack s;**

**s.push(3);**

**s.push(5);**

**s.getMin();**

**s.push(2);**

**s.push(1);**

**s.getMin();**

**s.pop();**

**s.getMin();**

**s.pop();**

**s.peek();**

**return 0;**

**}**