Computer Engineering Department, S.V.N.I.T. Surat.

B Tech (CO) –IInd Year semester-III

Course: *Data Structures CO203*

**Assignment-VI**

Q1.) Implement double ended priority queue [DeQueue]

The library is managing request for Book in first come first serve manner. Teacher's requests are given higher priority than that of students. Design the task of library using the most suitable Data Structure.

***Code:***

*// Implementation of "Output Restricted Deque"*

*// Insertion from Both Ends, But Deletion from Only One End Based on Priority*

*#include* <stdio.h>

*#include* <conio.h>

*#define* MAX 100

int deque[MAX];

int left = -1, right = -1;

void insert\_right(int);

void insert\_left(int);

void delete\_right(void);

void delete\_left(void);

void display(void);

*// Teacher is Inserted from Left & Student is Inserted from Right*

*// Student Request Time Array*

int stime[MAX];

*// Teachers Request Time Array*

int ttime[MAX];

int main()

{

    int i;

    int stud\_req = 0;

    int mxn = -1;

    printf("\nEnter the Number of Student Request : ");

    scanf("%d", &stud\_req);

*for* (i = 0; i < stud\_req; i++)

    {

        printf("Enter Student %d Request Time : ", i + 1);

        scanf("%d", &stime[i]);

*if* (stime[i] > mxn)

            mxn = stime[i];

    }

    int teah\_req = 0;

    printf("\nEnter the Number of Teacher Request : ");

    scanf("%d", &teah\_req);

*for* (i = 0; i < teah\_req; i++)

    {

        printf("Enter Teacher %d Request Time : ", i + 1);

        scanf("%d", &ttime[i]);

*if* (ttime[i] > mxn)

            mxn = ttime[i];

    }

*//Considering Both Student and Teacher Request Time to be Sorted*

*//Student is Inserted from Right*

*for* (i = stud\_req - 1; i >= 0; i--)

    {

        insert\_right(stime[i]);

    }

*// Teacher is Inserted from Left*

*for* (i = teah\_req - 1; i >= 0; i--)

    {

        insert\_left(ttime[i]);

    }

*// display();*

    int cur\_time;

*// Total Elements in Queue*

    int popcnt = stud\_req + teah\_req;

    int tcnt = 1;

    int scnt = 1;

    printf("Order of Execution of Request : \n\n");

*if* (stud\_req <= 0 && teah\_req <= 0)

    {

*if* (stud\_req == 0 && teah\_req == 0)

        {

            printf("No Request to be Processed!\n");

        }

*else*

        {

            printf("Invalid Input!\n");

        }

*return* 0;

    }

*while* (popcnt != 0)

    {

*// Teacher Request Time*

        int t = deque[left];

*// Student Request Time*

        int s = deque[right];

*if* (t == s)

        {

*// POP Teacher*

*if* (tcnt <= teah\_req)

            {

                printf("Teacher %d\n", tcnt);

                tcnt++;

                delete\_left();

            }

*else*

            {

                printf("Student %d\n", scnt);

                scnt++;

                delete\_right();

            }

        }

*else*

        {

*if* (t < s && tcnt <= teah\_req)

            {

*// POP Teacher*

                printf("Teacher %d\n", tcnt);

                tcnt++;

                delete\_left();

            }

*else*

            {

*// s < t && scnt<=stud\_req*

*// POP Student*

                printf("Student %d\n", scnt);

                scnt++;

                delete\_right();

            }

        }

        popcnt--;

    }

*return* 0;

}

*//-------INSERT AT RIGHT-------*

void insert\_right(int val)

{

*if* ((left == 0 && right == MAX - 1) || (left == right + 1))

    {

        printf("\nOVERFLOW");

*return*;

    }

*if* (left == -1) *//if queue is initially empty*

    {

        left = 0;

        right = 0;

    }

*else*

    {

*if* (right == MAX - 1)

            right = 0;

*else*

            right = right + 1;

    }

    deque[right] = val;

}

*//-------INSERT AT LEFT-------*

void insert\_left(int val)

{

*if* ((left == 0 && right == MAX - 1) || (left == right + 1))

    {

        printf("\nOVERFLOW");

*return*;

    }

*if* (left == -1) *//if queue is initially empty*

    {

        left = 0;

        right = 0;

    }

*else*

    {

*// To Avoid -ve Index in Array*

*// We Insert from Back of deque*

*if* (left == 0)

            left = MAX - 1;

*else*

            left = left - 1;

    }

    deque[left] = val;

}

*//-------DELETE FROM RIGHT-------*

void delete\_right()

{

*if* (left == -1)

    {

        printf("\nUNDERFLOW");

*return*;

    }

*// printf("\nThe deleted element is %d\n", deque[right]);*

*// int d = deque[right];*

*//Queue has only one element*

*if* (left == right)

    {

        left = -1;

        right = -1;

    }

*else*

    {

*if* (right == 0)

            right = MAX - 1;

*else*

            right = right - 1;

    }

*return*;

}

*//-------DELETE FROM LEFT-------*

void delete\_left()

{

*if* (left == -1)

    {

        printf("\nUNDERFLOW");

*return*;

    }

*// printf("\nThe deleted element is %d\n", deque[left]);*

*// int d = deque[left];*

*if* (left == right) *//Queue has only one element*

    {

        left = -1;

        right = -1;

    }

*else*

    {

*if* (left == MAX - 1)

            left = 0;

*else*

            left = left + 1;

    }

*return*;

}

*//-------DISPLAY-------*

void display()

{

    int front = left, rear = right;

*if* (front == -1)

    {

        printf("\nQueue is Empty\n");

*return*;

    }

    printf("\nDeQueue Elements : ");

*if* (front <= rear)

    {

*while* (front <= rear)

        {

            printf("%d ", deque[front]);

            front++;

        }

    }

*else*

    {

*while* (front <= MAX - 1)

        {

            printf("%d ", deque[front]);

            front++;

        }

        front = 0;

*while* (front <= rear)

        {

            printf("%d ", deque[front]);

            front++;

        }

    }

    printf("\n");

}

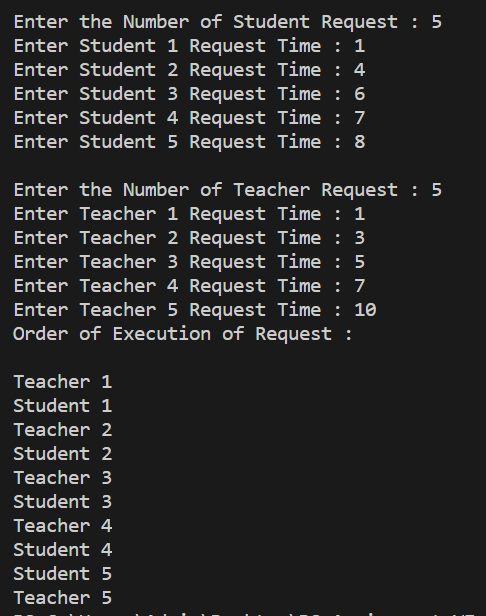
***Sample Input:***

(Assuming execution of every request is 1 sec)

Student request at time in seconds: 1,4,6,7,8

Teacher's request at time in seconds: 1,3,5,7,10

***Sample Output*** : [T1->S1-> T2->S2-> T3->S3-> T4->S4-> S5->T5]



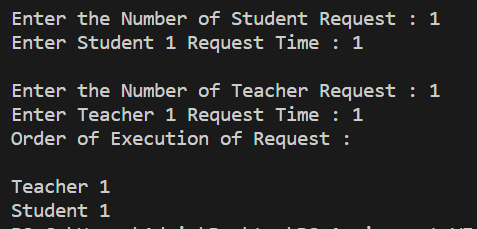
***Sample Input:***

(Assuming execution of every request is 1 sec)

Student request at time in seconds: 1

Teacher's request at time in seconds: 1

***Sample Output*** : [T1->S1]



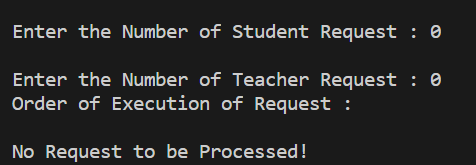
***Sample Input:***

(Assuming execution of every request is 1 sec)

Student request at time in seconds:

Teacher's request at time in seconds:

***Sample Output*** : [No Request]



***Sample Input:***

(Assuming execution of every request is 1 sec)

Student request at time in seconds: 2,3

Teacher's request at time in seconds: 1,3,5

***Sample Output*** : [T1->S1-> T2->S2-> T3]

