**DAILY ONLINE ACTIVITIES SUMMARY**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Date:** | **25-06-2020** | | | | | **Name:** | | **Apeksha Rane** | |
| **Sem & Sec** | **6th  Sem ‘A’ Sec** | | | | | **USN:** | | **4AL17CS010** | |
| **Online Test Summary** | | | | | | | | | |
| **Subject** | | **JSP and Data structures in C .** | | | | | | | |
| **Max. Marks** | | **JSP Quiz1=10.** | | **Score** | | | | **JSP Quiz1=Not attended** | |
| **Pre-Placement Training Summary** | | | | | | | | | |
| **Course** | **Workshop of JSP and Data structures in C .** | | | | | | | | |
| **Faculty** | | | **Sayeesh Sir.**  **Venkatesh Sir.** | | | | **Duration** | | **4 hours** |
| **Coding Challenges** | | | | | | | | | |
| **Problem Statement:** 1.write a c program to implement simple queue using SLL.  2. write a c program to implement circular queue using SLL.  3. write a c program to implement Multiple stack operations using SLL.  4. write a c program to implement operations of multiple queues. | | | | | | | | | |
| **Status: done** | | | | | | | | | |
| **Uploaded the report in Github** | | | | | **Yes** | | | | |
| **If yes Repository name** | | | | | Daily Report =<https://github.com/Apeksha12appu/19-5-2020-online-coding-activity> | | | | |
| **Uploaded the report in slack** | | | | | **Yes** | | | | |

**Class and Quiz Snapshots:**

**JSP and Data structures in C :**

**Not Attended**

**Coding Challenge:**

1.write a c program to implement simple queue using SLL.

#include <stdio.h>

// A C program to demonstrate linked list based implementation of queue

#include <stdio.h>

#include <stdlib.h>

struct QNode {

int key;

struct QNode\* next;

};

struct Queue {

struct QNode \*front, \*rear;

};

struct QNode\* newNode(int k)

{

struct QNode\* temp = (struct QNode\*)malloc(sizeof(struct QNode));

temp->key = k;

temp->next = NULL;

return temp;

}

struct Queue\* createQueue()

{

struct Queue\* q = (struct Queue\*)malloc(sizeof(struct Queue));

q->front = q->rear = NULL;

return q;

}

void enQueue(struct Queue\* q, int k)

{

struct QNode\* temp = newNode(k);

if (q->rear == NULL) {

q->front = q->rear = temp;

return;

}

q->rear->next = temp;

q->rear = temp;

}

void deQueue(struct Queue\* q)

{

if (q->front == NULL)

return;

struct QNode\* temp = q->front;

q->front = q->front->next;

if (q->front == NULL)

q->rear = NULL;

free(temp);

}

int main()

{

struct Queue\* q = createQueue();

enQueue(q, 10);

enQueue(q, 20);

deQueue(q);

deQueue(q);

enQueue(q, 30);

enQueue(q, 40);

enQueue(q, 50);

deQueue(q);

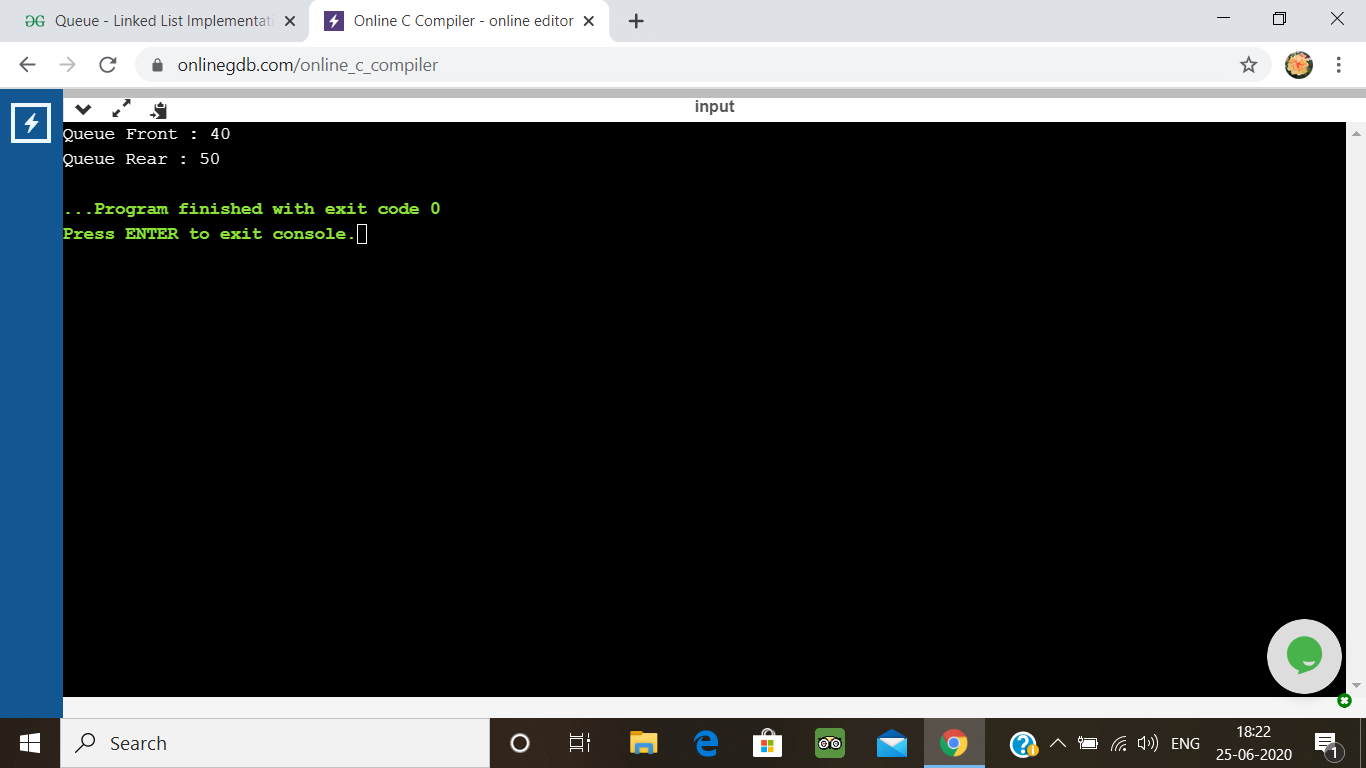
printf("Queue Front : %d \n", q->front->key);

printf("Queue Rear : %d", q->rear->key);

return 0;

}

**Output:**



2. write a c program to implement circular queue using SLL.

#include<stdio.h>

#include<stdlib.h>

typedef struct node{

int data;

struct node \*next;

}node;

node \*createQueue(){

node \*rear=NULL,\*front=NULL;

printf("How many items you want to create in queue?");

int n,data;

scanf("%d",&n);

int i = 0;

while(i<n){

if(i==0){

printf("Enter the data: ");

scanf("%d",&data);

node \*newNode = (node \*)malloc(sizeof(node));

newNode->data = data;

newNode->next=NULL;

rear = newNode;

front = rear;

}else{

printf("Enter the data: ");

scanf("%d",&data);

node \*newNode = (node \*)malloc(sizeof(node));

newNode->data = data;

newNode->next=front;

rear->next=newNode;

rear = newNode;

}

++i;

}

return rear;

}

void display(node \*rear, node \*front){

if(!rear){

printf("\nNOTHING TO DISPLAY\n");

return;

}

node \*temp = front;

if(!rear->next){

printf("%d\n",rear->data);

return;

}

if(temp!=rear){

printf("%d ",temp->data);

display(rear, temp->next);

}else{

printf("%d\n",temp->data);

}

}

void enqueue(node \*\*rear,int data){

node \*newNode = (node \*)malloc(sizeof(node));

newNode->data = data;

if(!(\*rear)){

newNode->next=NULL;

\*rear = newNode;

return;

}

node \*temp = \*rear;

if(!(\*rear)->next){

newNode->next = temp;

}else{

newNode->next = temp->next;

}

temp->next = newNode;

\*rear = newNode;

}

int dequeue(node \*rear){

int data;

if(!rear){

printf("\nNO ITEMS IN THE QUEUE\n");

return 0;

}

if(!rear->next){

data = rear->data;

free(rear);

return data;

}

node \*temp = rear->next;

data = temp->data;

rear->next = temp->next;

free(temp);

return data;

}

int main(){

node \*rear = createQueue();

display(rear,rear->next);

enqueue(&rear,99);

display(rear,rear->next);

int data = dequeue(rear);

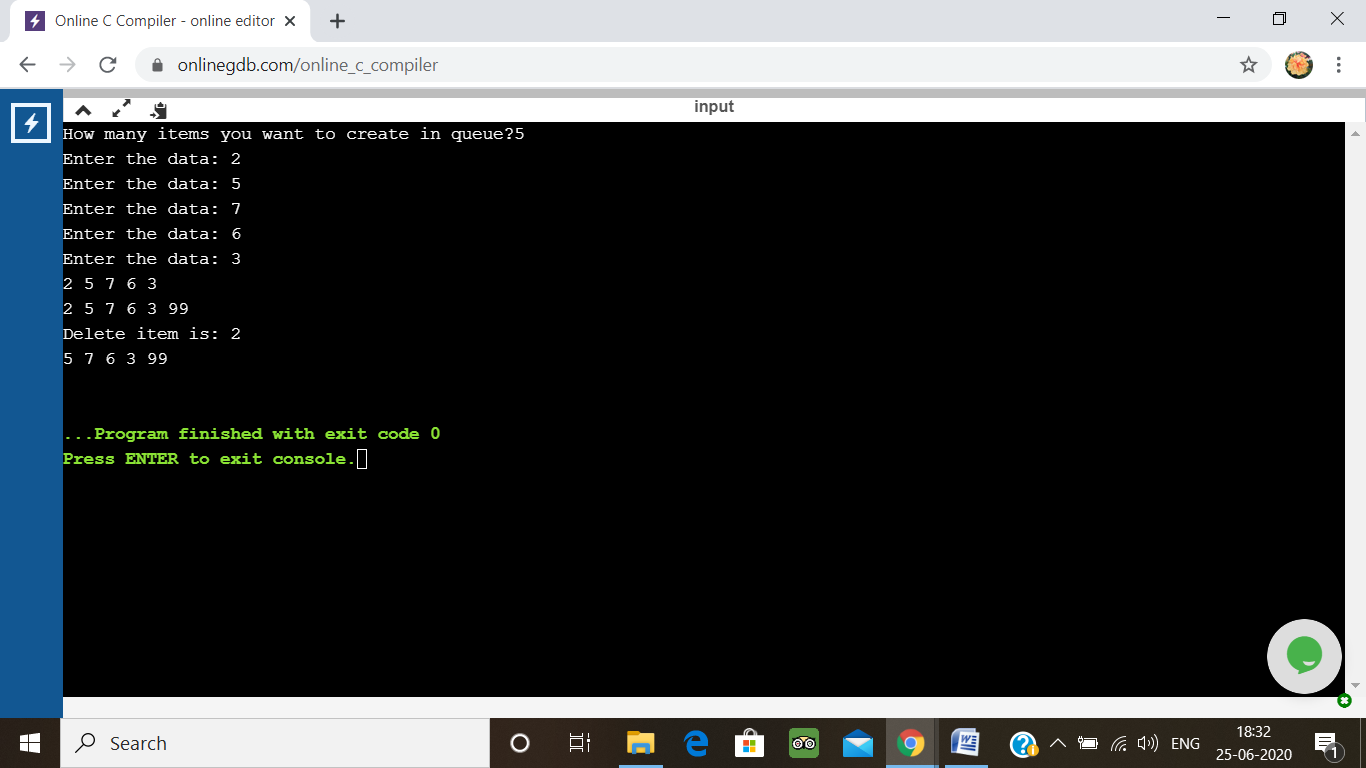
printf("Delete item is: %d\n",data);

display(rear,rear->next);

return 0;

}

**Output:**



3. write a c program to implement Multiple stack operations using SLL.

#include<stdio.h>

#include<conio.h>

#define MAX\_X 5

#define MAX\_Y 5

int topx=-1;

int topy=10;

void push\_x(int \*stack)

{

int info;

if(topx>=(MAX\_X-1))

{ printf("\n\nStack OverFlow");

return;

}

else

{ printf("\n\nEnter The info To Push");

scanf("%d",&info);

topx++;

stack[topx]=info;

}}

void push\_y(int \*stack)

{

int info;

if(topy<=(MAX\_Y))

{

printf("\n\nStack OverFlow");

return;

}

else

{

printf("\n\nEnter The info To Push");

scanf("%d",&info);

topy--;

stack[topy]=info;

}

}

void pop\_x(int \*stack)

{ if(topx==-1)

{

printf("Stack X is Underflow");

return;

}

else

{

printf("Item Poped from stack X is:%d\n",stack[topx]);

topx--;

}

}

void pop\_y(int \*stack)

{ if(topy==10)

{printf("Stack y is Underflow");

return;

}

else

{ printf("Item Poped from stack Y is:%d\n",stack[topy]);

topy++;

}}

void display\_x(int \*stack)

{

int i;

if(topx==-1)

{

printf("Stack X is Empty");

return;

}

else

{ for(i=topx;i>=0;i--)

{printf("%d,",stack[i]);}

printf("\n");

}}

void display\_y(int \*stack)

{

int i;

if(topy==10)

{printf("Stack Y is Empty");

return;}

else

{for(i=topy;i<=9;i++)

{

printf("%d,",stack[i]);

}

printf("\n");

} }

main()

{ int choice;

char ch;

int stack[MAX\_X+MAX\_Y];

do

{ printf("1.Push\_X\n2.Push\_Y\n");

printf("\n3.Pop\_X\n4.Pop\_Y\n");

printf("\n5.Display\_X\n6.Display\_Y\n");

printf("\n7.Exit");

printf("\n\nEnter Choice");

scanf("%d",&choice);

switch(choice)

{

case 1: push\_x(stack);break;

case 2: push\_y(stack);break;

case 3: pop\_x(stack);break;

case 4: pop\_y(stack);break;

case 5: display\_x(stack);break;

case 6: display\_y(stack);break;

case 7: break;

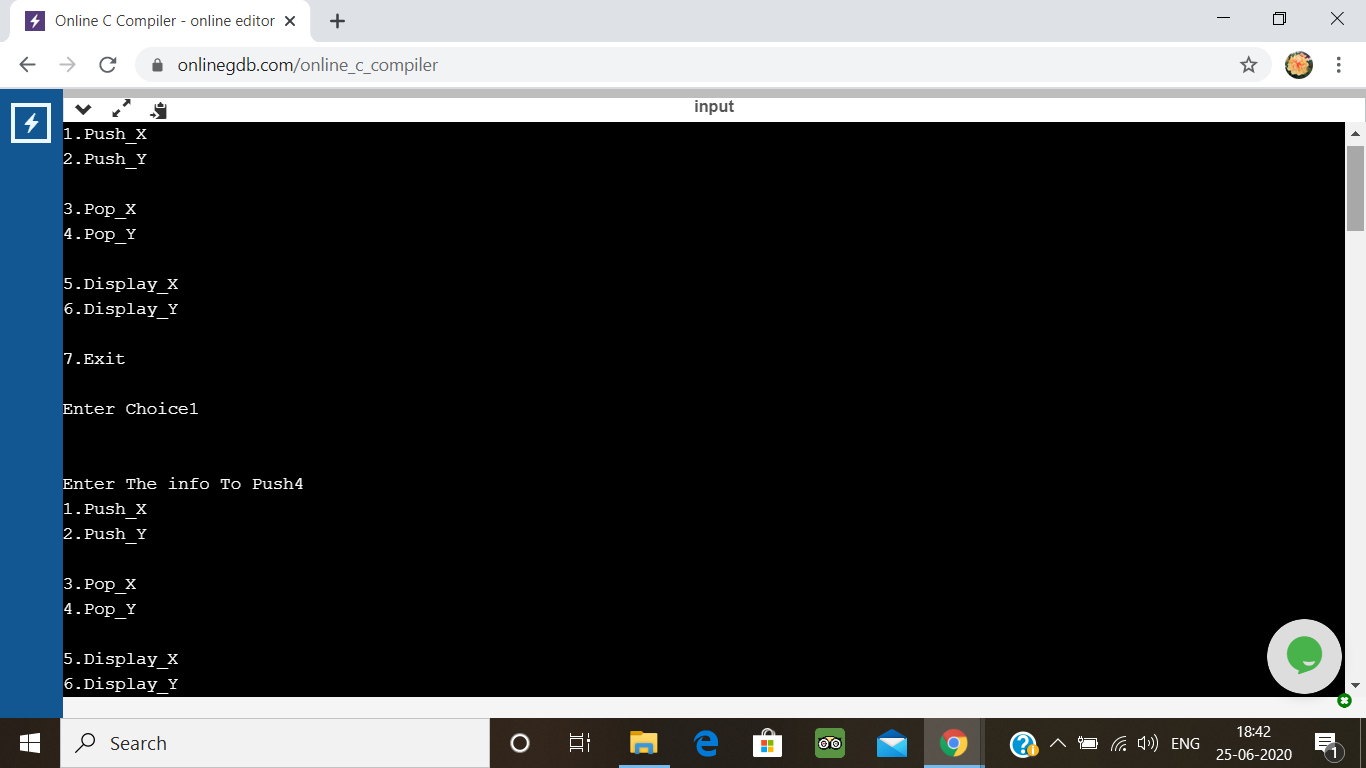
default: printf("Wrong Option...");

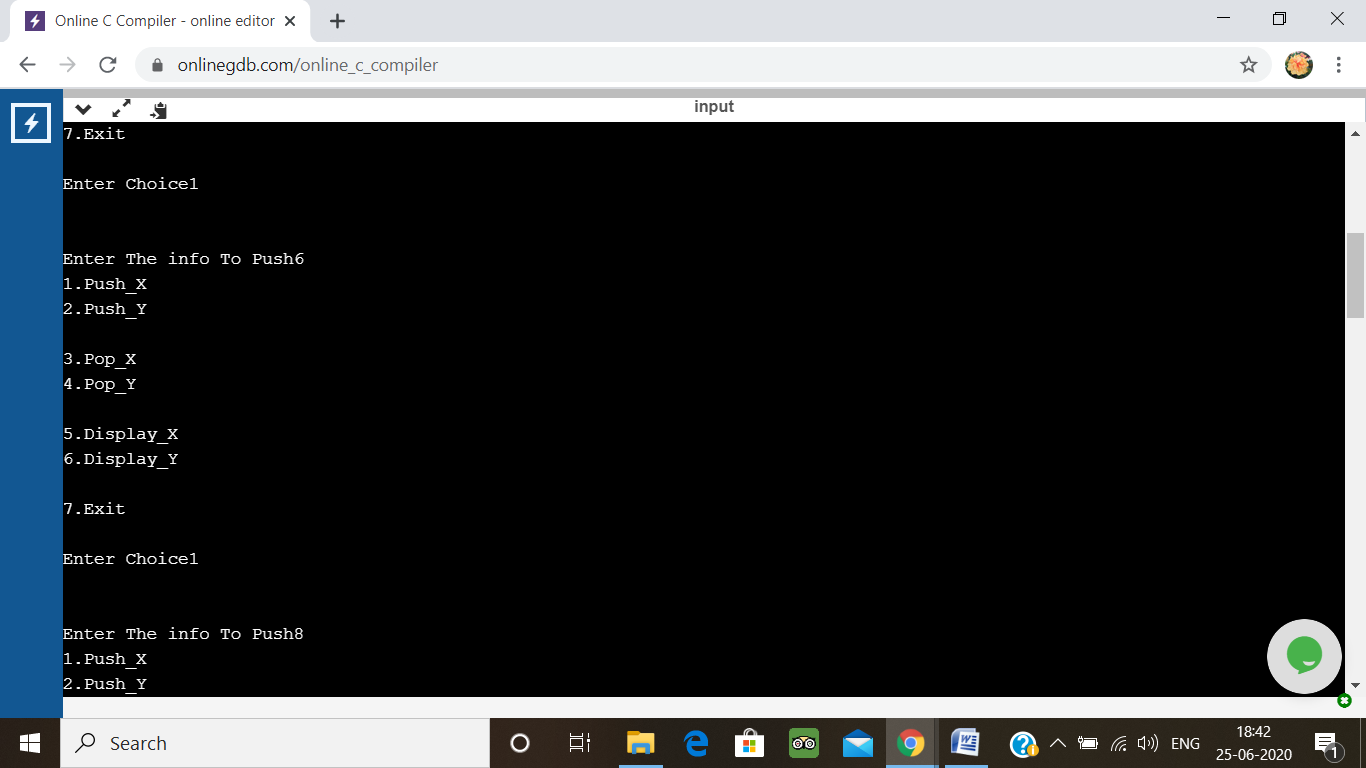
}

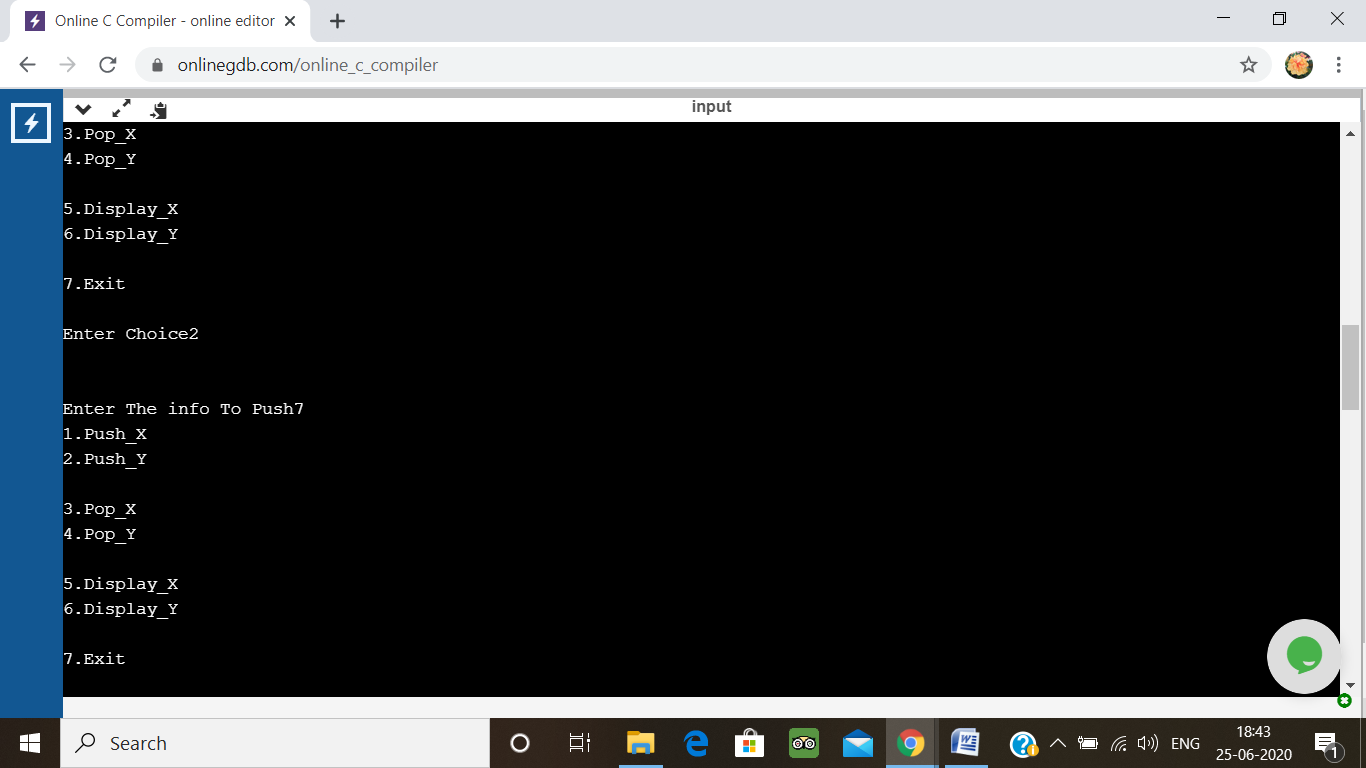
}while(choice!=7);

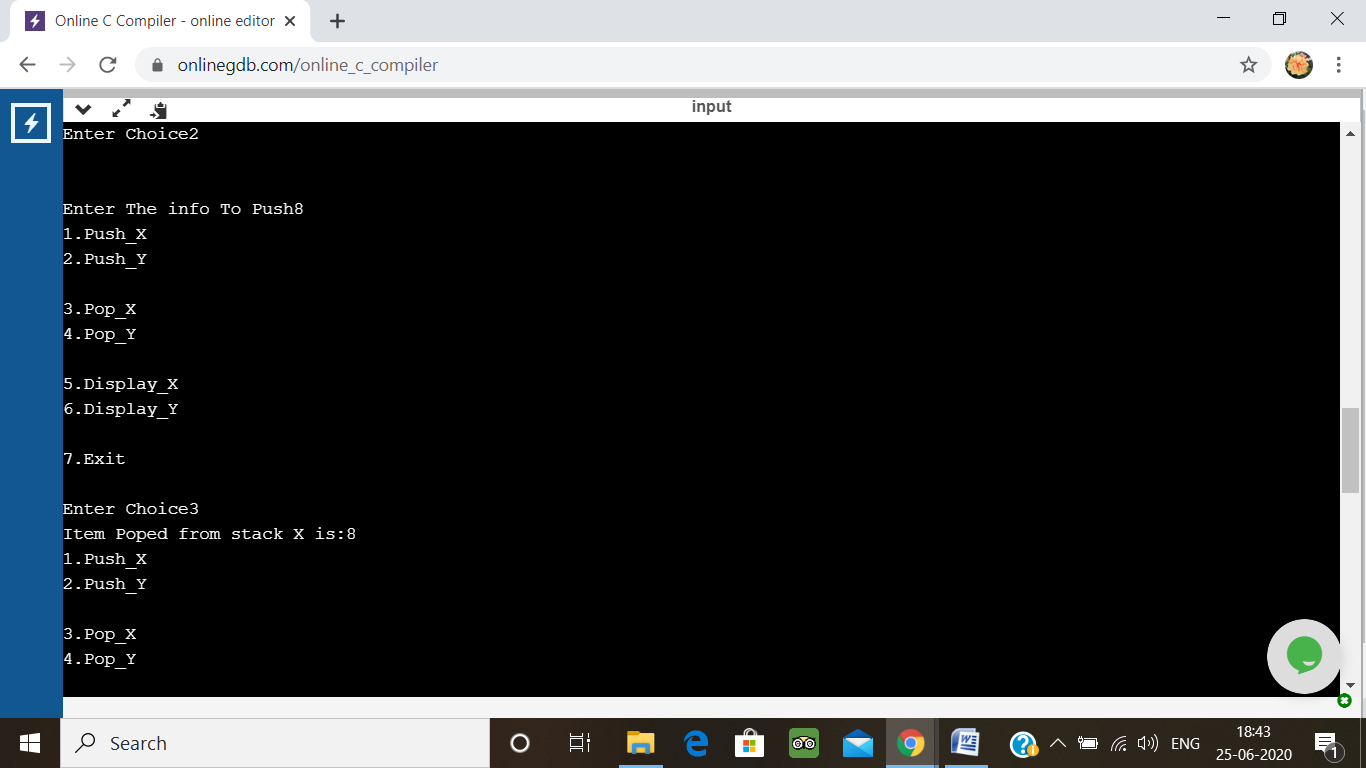
}

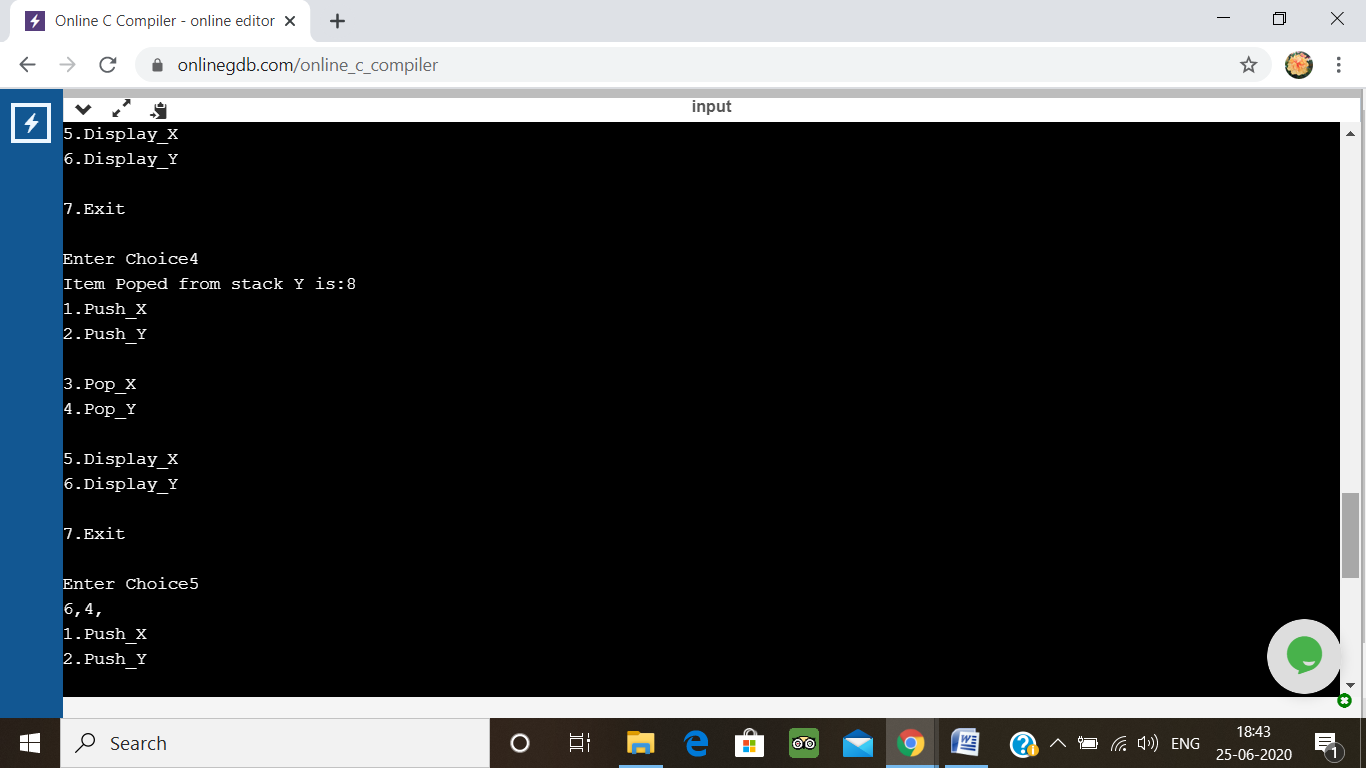
**Output:**

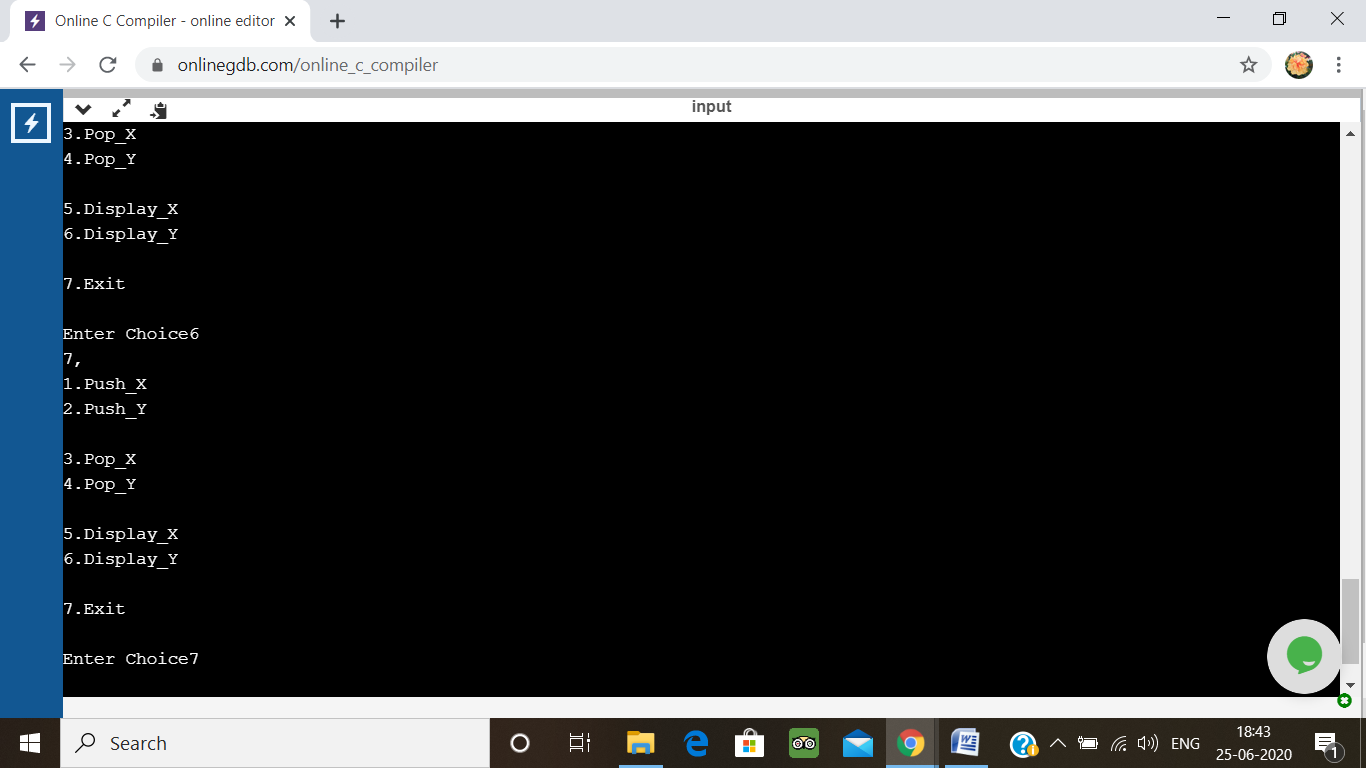












4. write a c program to implement operations of multiple queues.

#include<stdio.h>

#include<conio.h>

# define max 20

int insq (int queue[max], int qno, int rear[], int limit[], int \*data) {

if (rear[qno] == limit[qno])

return(-1);

else {

rear[qno]++;

queue[ rear[qno] ] = \*data;

return(1);

}

}

int delq (int queue[max], int qno, int front[], int rear[], int \*data) {

if( front[qno] == rear[qno] )

return(-1);

else {

front[qno]++;

\*data = queue[ front[qno] ];

return(1);

}

}

int getQueueNumber(int n) {

int qNo=0;

Inva:

printf("\n Enter a Logical Queue Number (1 to %d) : ", n);

scanf("%d", &qNo);

if (qNo<1 || qNo >n) {

printf(" Invalid Queue Number. Please try again.\n");

goto Inva;

}

return qNo;

}

void main() {

int queue[max], data;

int bott[10], limit[10], f[10], r[10];

int i, n, qno, size, option, reply;

printf("\n C Language program to implement the Multiple Queues \n");

printf("\n How Many Queues ? : ");

scanf("%d", &n);

size = max / n;

bott[0] = -1;

for(i = 1; i < n; i++)

bott[i] = bott[i-1] + size;

for(i = 0; i < n; i++)

limit[i] = bott[i] + size;

for(i = 0; i < n; i++)

f[i] = r[i] = bott[i];

do {

printf("\n\n C Language program to implement the Multiple Queues \n");

printf("\n 1. Insert in a Queue");

printf("\n 2. Delete from a Queue");

printf("\n 3. Print from a Queue");

printf("\n 3. Exit \n");

printf("\n Select proper option ( 1 / 2 / 3 / 4) : ");

scanf("%d", &option);

switch(option) {

case 1 : //... Insert

qno = getQueueNumber(n);

printf("\n Enter Data : ");

scanf("%d", &data);

reply = insq(queue, qno-1, r, limit, &data);

if( reply == -1)

printf("\n Queue %d is Full \n", qno);

else

printf("\n %d is inserted in a Queue No. %d \n", data, qno);

break;

case 2 : //... Delete

qno = getQueueNumber(n);

reply = delq(queue, qno-1, f, r, &data);

if( reply == -1)

printf("\n Queue %d is Empty \n", qno);

else

printf("\n %d is deleted from Queue No. %d \n", data, qno);

break;

case 3:

qno = getQueueNumber(n);

printf("\n Elements of Queue %d are as : ", qno);

if (f[qno-1]==r[qno-1]) {

printf("\n Queue is empty");

break;

}

for (i=f[qno-1]+1; i<=r[qno-1]; i++)

printf("%d\t", queue[i]);

printf("\n");

break;

case 4 :

break;

default:

printf("\n Invalid input. Please try again.");

}

}while(option!=4);

}

**Output:**

