**Programming Lab**

**Assignment 3**

**Question :-**

1. Write a program to implement a stack data structure using an array. 2. Implement a system which can handle more than one stack (n stacks) using arrays.

3. Write a Boolean function to return true if two stacks are equal.

4. Write a program for dynamic implementation (using link list) of stacks (n stacks).

5. Write a program to implement queue data structure using an array.

6. Implement a system which can handle more than one queue (n queues) using arrays.

7. A Boolean function to return true if two queues are equal.

8. Write a program for dynamic implementation (using link list) of a queues (n queues).

9. Implement a priority queue using linked list (any type)

**Program 1:-**

#include<stdio.h>

#include<stdlib.h>

void push(int \*arr,int \*top,int val,int n){

    if(\*top + 1 == n){

        printf("stack is full\n");

    }

    else{

        \*top = \*top + 1;

        arr[\*top] = val;

        printf("Element inserted\n");

    }

}

int pop(int \*arr,int \*top,int n){

    if(\*top == -1){

        printf("stack is empty");

        return -1;

    }

    else{

        int x = arr[\*top];

        \*top = \*top - 1;

        return x;

    }

}

void display(int arr[],int top){

    int i;

    for(i=0;i<=top;i++){

        printf("%d ",arr[i]);

    }

    printf("\n");

}

int main(){

    int n,top=-1;

    printf("Enter the maximum number of element you want in the stack\n");

    scanf("%d",&n);

    int arr[n],op,x;

    printf("1 for inserting an element\n2 for deleting an element\n3 for displaying the elements\n");

    while(1){

        printf("Enter your operation\n");

        scanf("%d",&op);

        switch (op)

        {

        case 1:

            printf("Enter the element to be inserted\n");

            scanf("%d",&x);

            push(arr,&top,x,n);

            break;

        case 2:

            x = pop(arr,&top,n);

            if(x >= 0){

                printf("%d has been deleted\n",x);

            }

            break;

        case 3:

            printf("Elements in the stack are\n");

            display(arr,top);

            break;

        default:

            exit(0);

        }

    }

}

**Program 2 and 3 :-**

#include<stdio.h>

#include<stdlib.h>

void push(int \*arr,int \*top,int val,int size){

    if(\*top + 1 == size){

        printf("stack is full\n");

    }

    else{

        \*top = \*top + 1;

        arr[\*top] = val;

        printf("Element inserted\n");

    }

}

void pop(int \*arr,int \*top){

    if(\*top == -1){

        printf("stack is empty");

    }

    else{

        \*top = \*top - 1;

        printf("Element deleted\n");

    }

}

void is\_same(int arr1[],int top1,int arr2[],int top2){

    if(top1 != top2){

        printf("These two stack are not same\n");

    }else{

        int flag = 1;

        for(int i=0;i<=top1;i++){

            if(arr1[i]!=arr2[i]){

                flag = 0;

                break;

            }

        }

        if(flag == 1){

            printf("These two stack are same\n");

        }else{

            printf("These two stack are not same\n");

        }

    }

}

void display(int arr[],int top){

    int i;

    for(i=0;i<=top;i++){

        printf("%d ",arr[i]);

    }

    printf("\n");

}

int main(){

    int n,size;

    printf("Enter the number of stacks to be created\n");

    scanf("%d",&n);

    printf("Enter the size of each stacks\n");

    scanf("%d",&size);

    int arr[n][size],top[n],op,x,no,n1,n2;

    for(int i=0;i<n;i++){

        top[i] = -1;

    }

    printf("1 for inserting an element\n2 for deleting an element\n3 for displaying the elements\n4 for comparing wo stacks\n");

    while(1){

        printf("Enter your operation\n");

        scanf("%d",&op);

        switch (op)

        {

        case 1:

            printf("Enter the stack no...\n");

            scanf("%d",&no);

            printf("Enter the element to be inserted\n");

            scanf("%d",&x);

            push(arr[no-1],&top[no-1],x,size);

            break;

        case 2:

            printf("Enter the stack no...\n");

            scanf("%d",&no);

            pop(arr[no-1],&top[no-1]);

            break;

        case 3:

            printf("Enter the stack no...\n");

            scanf("%d",&no);

            printf("Elements in the stack %d are\n",no);

            display(arr[no-1],top[no-1]);

            break;

        case 4:

            printf("Enter the 1st stack no...\n");

            scanf("%d",&n1);

            printf("Enter the 2nd stack no...\n");

            scanf("%d",&n2);

            is\_same(arr[n1-1],top[n1-1],arr[n2-1],top[n2-1]);

            break;

        default:

            exit(0);

        }

    }

}

**Program 4 :-**

#include<stdio.h>

#include<stdlib.h>

struct node

{

    int data;

    int pos;

    struct node \*next;

};

void push(struct node \*\*head,int val,int size){

    struct node \*newnode = (struct node \*)malloc(sizeof(struct node));

    newnode->data = val;

    struct node \*temp = \*head;

    if(temp == NULL){

        newnode->pos = 0;

        temp = newnode;

        temp->next = NULL;

        \*head = temp;

    } else{

        while(temp->next != NULL){

            temp = temp->next;

        }

        if(temp->pos + 1 == size){

                printf("stack is full\n");

        }else{

            newnode->pos = temp->pos + 1;

            temp->next = newnode;

            temp= newnode;

            temp->next = NULL;

        }

    }

    printf("Element inserted\n");

}

void pop(struct node \*\*head){

    struct node \*temp = \*head;

    if(temp == NULL){

        printf("stack is empty\n");

    }

    else{

        while(temp->next->next != NULL){

            temp = temp->next;

        }

        temp->next = NULL;

        printf("Element deleted\n");

    }

}

void display(struct node \*list){

    struct node \*temp = list;

    while(temp != NULL){

        printf("->%d",temp->data);

        temp = temp->next;

    }

    printf("\n");

}

int main(){

    int n,size;

    printf("Enter the number of stacks to be created\n");

    scanf("%d",&n);

    printf("Enter the size of each stack\n");

    scanf("%d",&size);

    struct node \*stacks[n];

    for(int i=0;i<n;i++){

        stacks[i] = NULL;

    }

    int op,x,no;

    printf("1 for inserting an element\n2 for deleting an element\n3 for displaying the elements\n");

    while(1){

        printf("Enter your operation\n");

        scanf("%d",&op);

        switch (op)

        {

        case 1:

            printf("Enter the stack no...\n");

            scanf("%d",&no);

            printf("Enter the element to be inserted\n");

            scanf("%d",&x);

            push(&stacks[no-1],x,size);

            break;

        case 2:

            printf("Enter the stack no...\n");

            scanf("%d",&no);

            pop(&stacks[no-1]);

            break;

        case 3:

            printf("Enter the stack no...\n");

            scanf("%d",&no);

            printf("Elements in the stack %d are\n",no);

            display(stacks[no-1]);

            break;

        default:

            exit(0);

        }

    }

}

**Program 5 :-**

#include<stdio.h>

#include<stdlib.h>

void push(int \*arr,int \*top,int val,int n){

    if(\*top + 1 == n){

        printf("queue is full\n");

    }

    else{

        \*top = \*top + 1;

        arr[\*top] = val;

        printf("Element inserted\n");

    }

}

int pop(int \*arr,int \*top,int n){

    if(\*top == -1){

        printf("queue is empty");

        return -1;

    }

    else{

        int x = arr[0];

        for(int i=1;i<=\*top;i++){

            arr[i-1] = arr[i];

        }

        \*top = \*top - 1;

        return x;

    }

}

void display(int arr[],int top){

    int i;

    for(i=0;i<=top;i++){

        printf("%d ",arr[i]);

    }

    printf("\n");

}

int main(){

    int n,top=-1;

    printf("Enter the maximum number of element you want in the queue\n");

    scanf("%d",&n);

    int arr[n],op,x;

    printf("1 for inserting an element\n2 for deleting an element\n3 for displaying the elements\n");

    while(1){

        printf("Enter your operation\n");

        scanf("%d",&op);

        switch (op)

        {

        case 1:

            printf("Enter the element to be inserted\n");

            scanf("%d",&x);

            push(arr,&top,x,n);

            break;

        case 2:

            x = pop(arr,&top,n);

            if(x >= 0){

                printf("%d has been deleted\n",x);

            }

            break;

        case 3:

            printf("Elements in the queue are\n");

            display(arr,top);

            break;

        default:

            exit(0);

        }

    }

}

**Program 6 and 7 :-**

#include<stdio.h>

#include<stdlib.h>

void push(int \*arr,int \*top,int val,int size){

    if(\*top + 1 == size){

        printf("queue is full\n");

    }

    else{

        \*top = \*top + 1;

        arr[\*top] = val;

        printf("Element inserted\n");

    }

}

void pop(int \*arr,int \*top){

    if(\*top == -1){

        printf("queue is empty");

    }

    else{

        int x = arr[0];

        for(int i=1;i<=\*top;i++){

            arr[i-1] = arr[i];

        }

        \*top = \*top - 1;

        printf("Element deleted\n");

    }

}

void is\_same(int arr1[],int top1,int arr2[],int top2){

    if(top1 != top2){

        printf("These two stack are not same\n");

    }else{

        int flag = 1;

        for(int i=0;i<=top1;i++){

            if(arr1[i]!=arr2[i]){

                flag = 0;

                break;

            }

        }

        if(flag == 1){

            printf("These two stack are same\n");

        }else{

            printf("These two stack are not same\n");

        }

    }

}

void display(int arr[],int top){

    int i;

    for(i=0;i<=top;i++){

        printf("%d ",arr[i]);

    }

    printf("\n");

}

int main(){

    int n,size;

    printf("Enter the number of queues to be created\n");

    scanf("%d",&n);

    printf("Enter the size of each queues\n");

    scanf("%d",&size);

    int arr[n][size],top[n],op,x,no,n1,n2;

    for(int i=0;i<n;i++){

        top[i] = -1;

    }

    printf("1 for inserting an element\n2 for deleting an element\n3 for displaying the elements\n4 for comparing wo queues\n");

    while(1){

        printf("Enter your operation\n");

        scanf("%d",&op);

        switch (op)

        {

        case 1:

            printf("Enter the queue no...\n");

            scanf("%d",&no);

            printf("Enter the element to be inserted\n");

            scanf("%d",&x);

            push(arr[no-1],&top[no-1],x,size);

            break;

        case 2:

            printf("Enter the queue no...\n");

            scanf("%d",&no);

            pop(arr[no-1],&top[no-1]);

            break;

        case 3:

            printf("Enter the queue no...\n");

            scanf("%d",&no);

            printf("Elements in the queue %d are\n",no);

            display(arr[no-1],top[no-1]);

            break;

        case 4:

            printf("Enter the 1st queue no...\n");

            scanf("%d",&n1);

            printf("Enter the 2nd queue no...\n");

            scanf("%d",&n2);

            is\_same(arr[n1-1],top[n1-1],arr[n2-1],top[n2-1]);

            break;

        default:

            exit(0);

        }

    }

}

**Program 8 :-**

#include<stdio.h>

#include<stdlib.h>

struct node

{

    int data;

    int pos;

    struct node \*next;

};

void push(struct node \*\*head,int val,int size){

    struct node \*newnode = (struct node \*)malloc(sizeof(struct node));

    newnode->data = val;

    struct node \*temp = \*head;

    if(temp == NULL){

        newnode->pos = 0;

        temp = newnode;

        temp->next = NULL;

        \*head = temp;

    } else{

        while(temp->next != NULL){

            temp = temp->next;

        }

        if(temp->pos + 1 == size){

                printf("Queue is full\n");

        }else{

            newnode->pos = temp->pos + 1;

            temp->next = newnode;

            temp= newnode;

            temp->next = NULL;

        }

    }

    printf("Element inserted\n");

}

void pop(struct node \*\*head){

    struct node \*temp = \*head;

    if(temp == NULL){

        printf("Queue is empty\n");

    }

    else{

        temp = temp->next;

        \*head = temp;

        while(temp != NULL){

            temp->pos = temp->pos - 1;

            temp = temp->next;

        }

        printf("Element deleted\n");

    }

}

void display(struct node \*list){

    struct node \*temp = list;

    while(temp != NULL){

        printf("->%d",temp->data);

        temp = temp->next;

    }

    printf("\n");

}

int main(){

    int n,size;

    printf("Enter the number of queues to be created\n");

    scanf("%d",&n);

    printf("Enter the size of each queue\n");

    scanf("%d",&size);

    struct node \*queues[n];

    for(int i=0;i<n;i++){

        queues[i] = NULL;

    }

    int op,x,no;

    printf("1 for inserting an element\n2 for deleting an element\n3 for displaying the elements\n");

    while(1){

        printf("Enter your operation\n");

        scanf("%d",&op);

        switch (op)

        {

        case 1:

            printf("Enter the queue no...\n");

            scanf("%d",&no);

            printf("Enter the element to be inserted\n");

            scanf("%d",&x);

            push(&queues[no-1],x,size);

            break;

        case 2:

            printf("Enter the queue no...\n");

            scanf("%d",&no);

            pop(&queues[no-1]);

            break;

        case 3:

            printf("Enter the queue no...\n");

            scanf("%d",&no);

            printf("Elements in the queue %d are\n",no);

            display(queues[no-1]);

            break;

        default:

            exit(0);

        }

    }

}

**Program 9 :-**

#include<stdio.h>

#include<stdlib.h>

struct node{

    int data;

    int priority;

    struct node \*next;

};

struct node \*head = NULL;

int size=0,max\_size;

void push(int val,int priority){

    if(size == max\_size){

        printf("Queue is full...\n");

    }else{

        struct node \*newnode = (struct node \*)malloc(sizeof(struct node));

        newnode->data = val;

        newnode->priority =priority;

        newnode->next = NULL;

        if(head == NULL){

            head = newnode;

        }else{

            if(head->priority < priority){

                newnode->next = head;

                head = newnode;

                size++;

            }else{

                struct node \*temp = head;

                while(temp->next && temp->next->priority > priority){

                    temp = temp->next;

                }

                newnode->next = temp->next;

                temp->next = newnode;

                size++;

            }

        }

        printf("Element inserted...\n");

    }

}

void pop(){

    if(size == 0){

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

    }else{

        struct node \*temp = head;

        head = head->next;

        free(temp);

        size--;

        printf("Element deleted...\n");

    }

}

void display(){

    struct node \*temp = head;

    while(temp != NULL){

        printf("->%d",temp->data);

        temp = temp->next;

    }

    printf("\n");

}

int main(){

    printf("Enter the max size of the priority queue\n");

    scanf("%d",&max\_size);

    int op,val,priority;

    printf("1 for inserting an element\n2 for deleting an element\n3 for displaying the elements\n");

    while(1){

        printf("Enter your operation\n");

        scanf("%d",&op);

        switch (op)

        {

        case 1:

            printf("Enter the element to be inserted\n");

            scanf("%d",&val);

            printf("Enter the priority...\n");

            scanf("%d",&priority);

            push(val,priority);

            break;

        case 2:

            pop();

            break;

        case 3:

            printf("Elements in the queue are...\n");

            display();

            break;

        default:

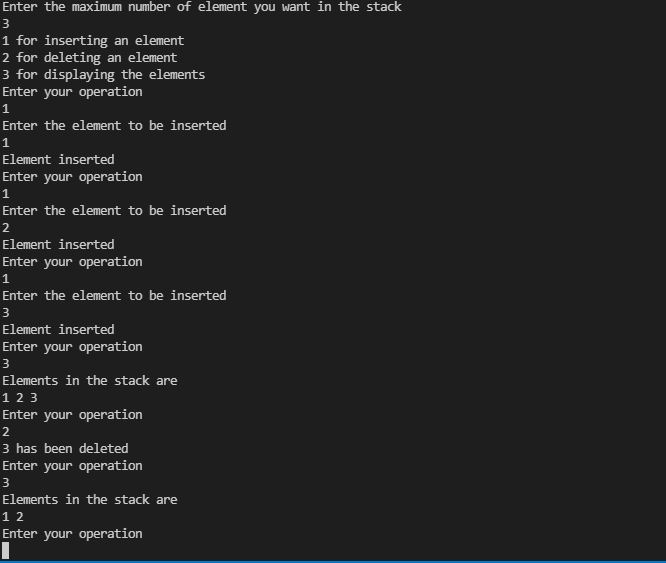
            exit(0);

        }

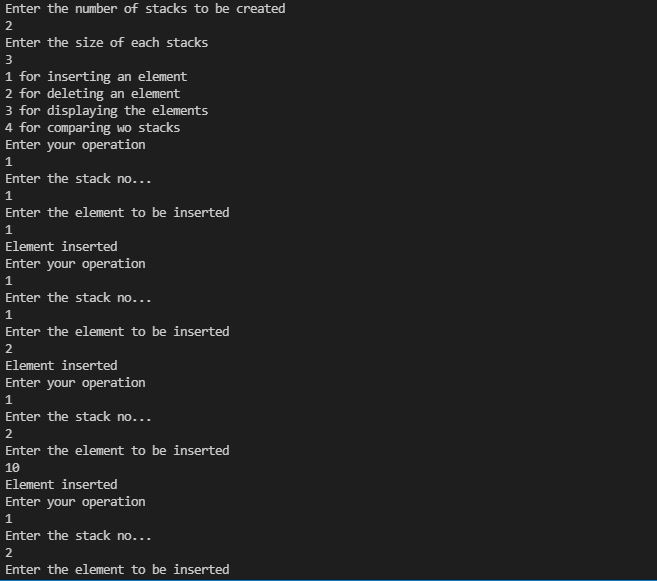
    }

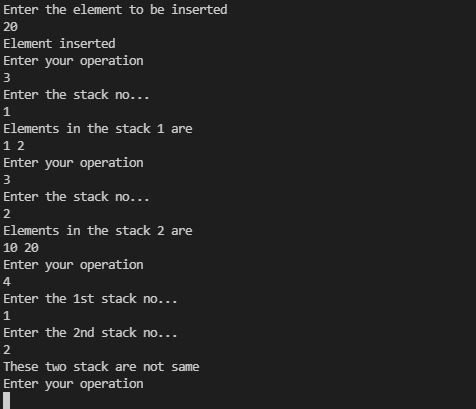
}

**Output 1 :-**

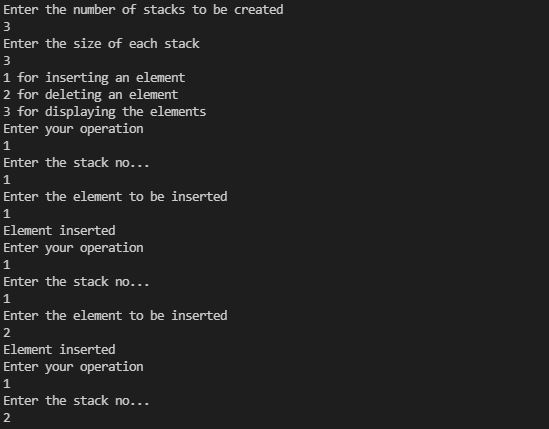
****

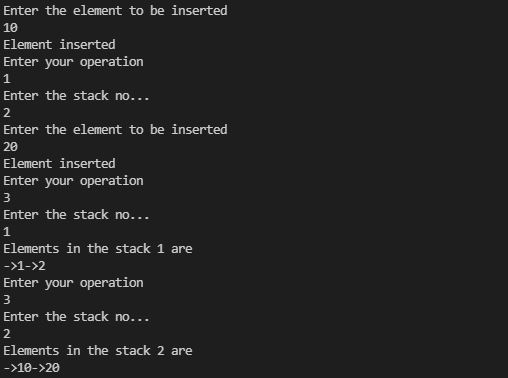
**Output 2 and 3 :-**

****

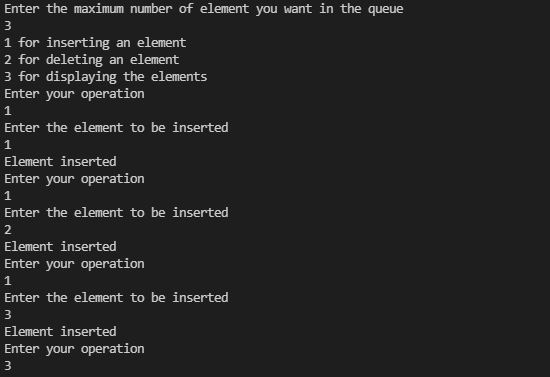


**Output 4 :-**

****

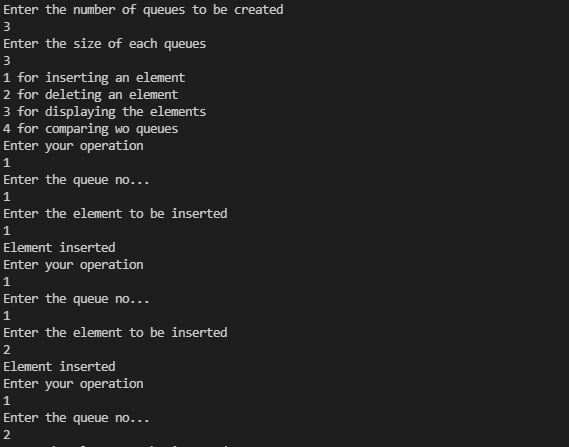


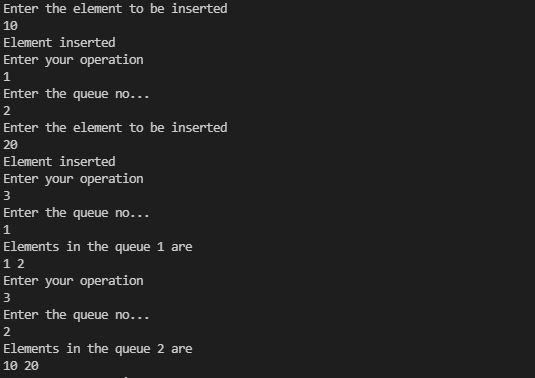
**Output 5 :-**

****

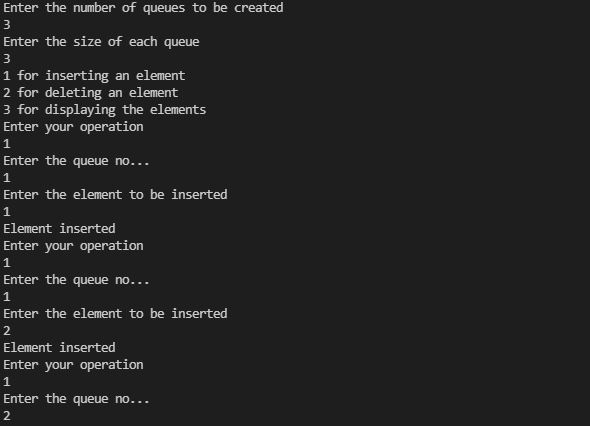


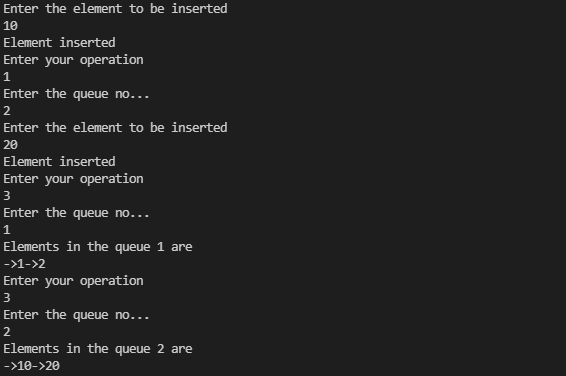
**Output 6 and 7 :-**

****

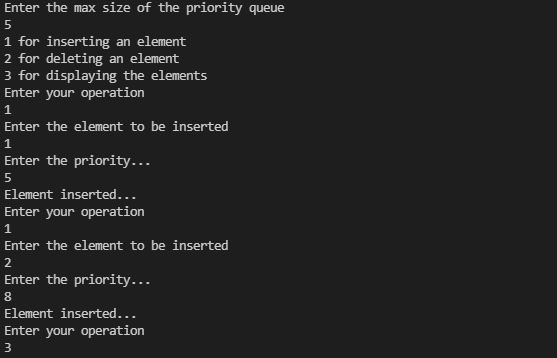


**Output 8 :-**

****



**Ouput 9 :-**

****

