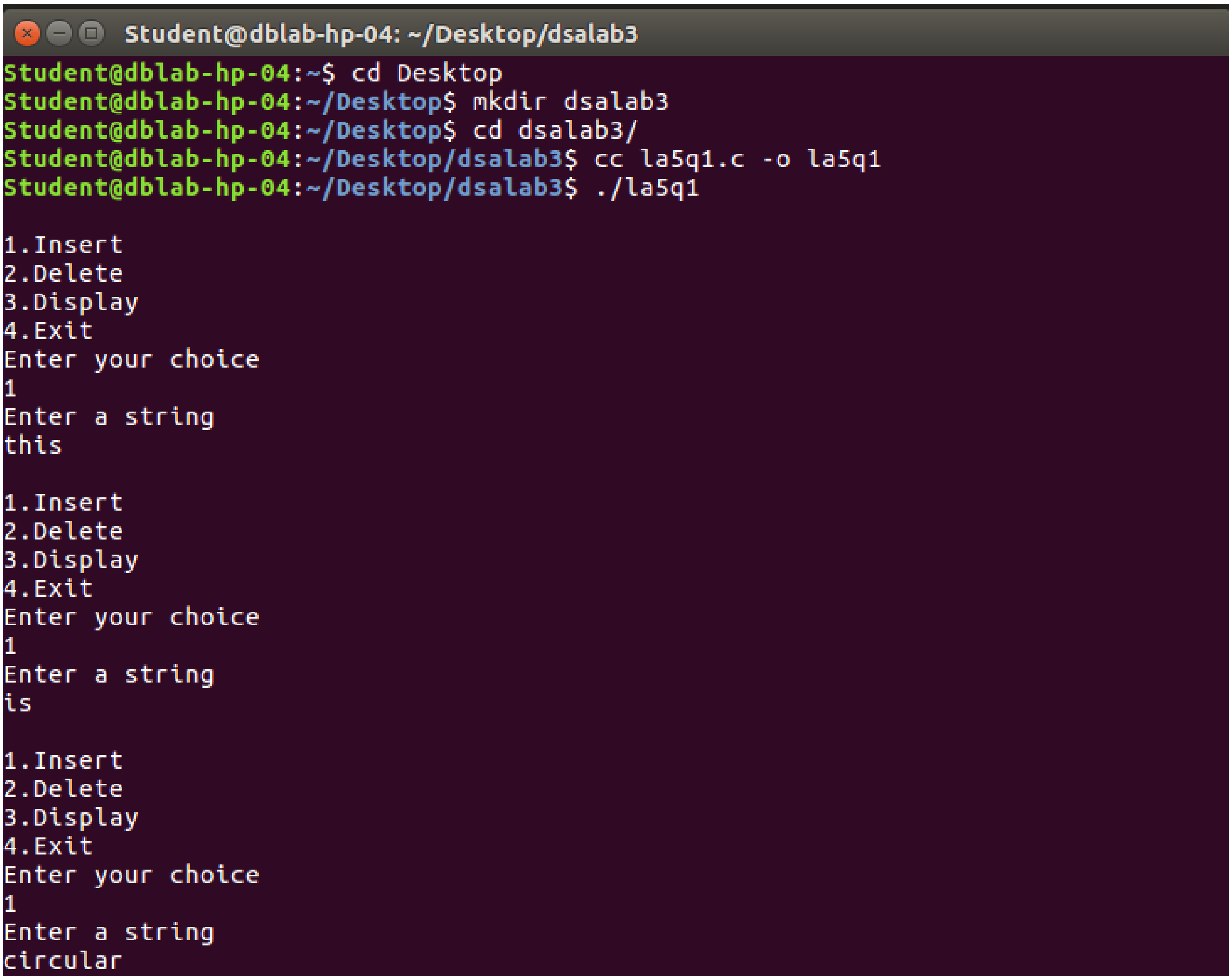
Ayush Goyal
190905522

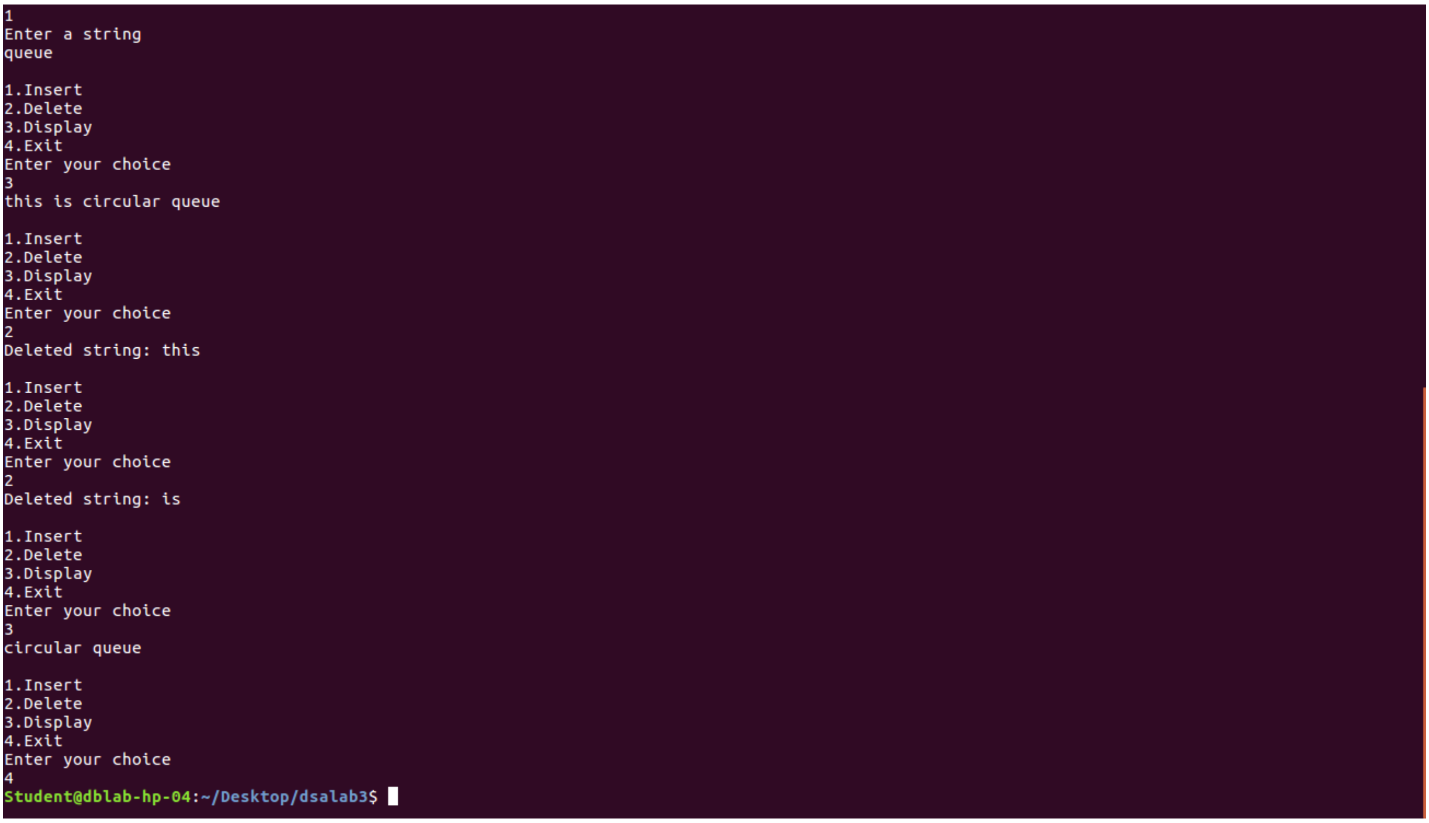
DSA Lab 5

Q1) Implement a circular queue of Strings using structures. Include functions insertcq, deletecq and displaycq.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define max_size 10
#define max_str 20

typedef struct{
   char **arr;
   int front,rear;
}QUE;

void initialize(QUE *cq){
   int i;
   cq->front = -1;
   cq->rear = -1;
   cq->arr = malloc(sizeof(char*)*max_size);
   for(i=0;i<max_size;i++){
      cq->arr[i] = malloc(sizeof(char)*max_str);
   }
}

void insertcq(QUE *cq,char *str){
   if(cq->front == cq->rear && cq->rear == -1){
      cq->rear=cq->front=0;
      strcpy(cq->arr[cq->rear],str);
      return;
   }
   if(cq->front == ((cq->rear)+1)%max_size){
      printf("Queue is full\n");
      return;
   }
   cq->rear = ((cq->rear)+1)%max_size;
   strcpy(cq->arr[cq->rear],str);
}

void deletecq(QUE *cq){
   char *ele;
   if(cq->front == cq->rear){
      printf("Queue underflow\n");
      return;
   }
   else{
        ele = cq->arr[cq->front];
        printf("Deleted string: %s\n", ele);
        cq->front=((cq->front)+1)%max_size;
    }
}

void display(QUE *cq){
    int i;
    if(cq->rear == cq->front){
        printf("Queue is empty\n");
        return;
    }
    else{
        for(i=cq->front;i!=cq->rear;i=(i+1)%max_size){
            printf("%s ",cq->arr[i]);
        }
        printf("%s\n", cq->arr[i]);
    }
}

int main(){
    QUE cq;
    initialize(&cq);
    int ch;
    char x[max_str];
    do{
        printf("\n1.Insert\n2.Delete\n3.Display\n4.Exit\n");
        printf("Enter your choice\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
                    printf("Enter a string\n");
                    scanf("%s",x);
                    insertcq(&cq,x);
                    break;
            case 2:
                    deletecq(&cq);
                    break;
            case 3:
                    display(&cq);
                    break;
            case 4:
                    exit(5);
        }
    }while(ch!=4);
    return 0;
}
```



```
Student@dblab-hp-04:~$ cd Desktop
Student@dblab-hp-04:~/Desktop$ mkdir dsalab3
Student@dblab-hp-04:~/Desktop$ cd dsalab3/
Student@dblab-hp-04:~/Desktop/dsalab3$ cc la5q1.c -o la5q1
Student@dblab-hp-04:~/Desktop/dsalab3$ ./la5q1

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice
1
Enter a string
this

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice
1
Enter a string
is

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice
1
Enter a string
circular
```



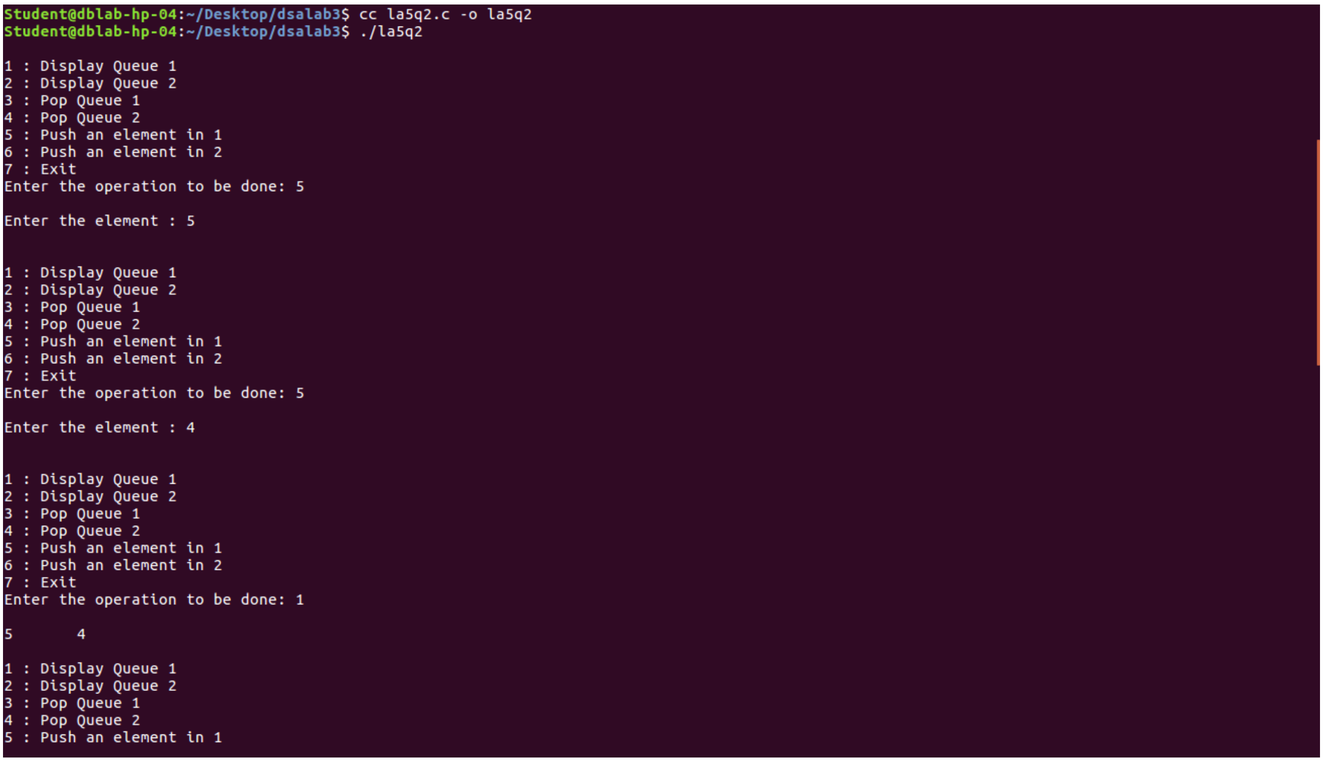
```
1
Enter a string
queue

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice
3
this is circular queue

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice
2
Deleted string: this

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice
2
Deleted string: is

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice
3
circular queue

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice
4
Student@dblab-hp-04:~/Desktop/dsalab3$ 
```

Q2)Implement two circular queues of integers in a single array where first queue will run from 0 to N/2 and second queue will run from N/2+1 to N-1 where N is the size of the array.

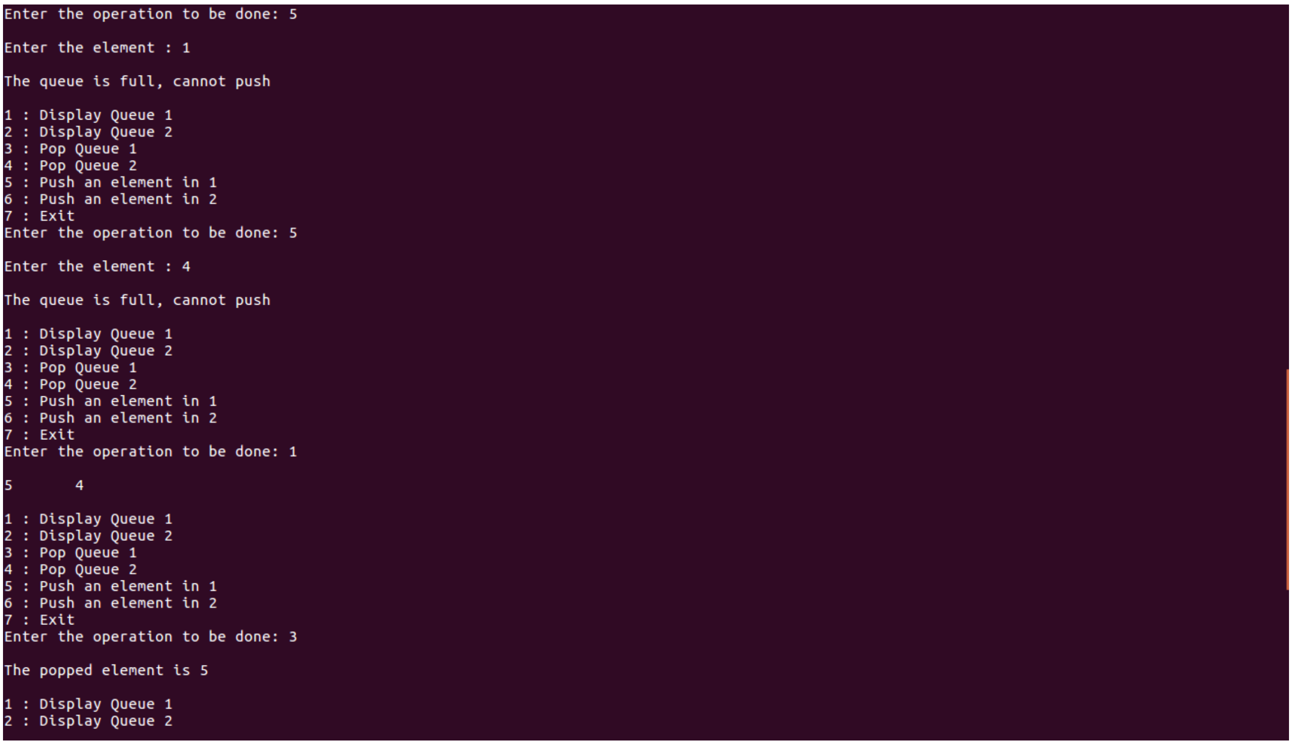
```
#include <stdio.h>
#include <stdlib.h>
#define MAX_QUEUE_SIZE1 3
#define MAX_QUEUE_SIZE2 3
#define MAX_QUEUE_SIZE 6



typedef struct
{
    int front1,rear1,front2,rear2;
    int array[MAX_QUEUE_SIZE1 + MAX_QUEUE_SIZE2];
} Queue;

void display1(Queue q)
{
    if (q.front1 == q.rear1)
    {
        printf("\nThe queue is empty, nothing to print");
    }
    else
    {
        printf("\n");
        for (int i = (q.front1 + 1) % MAX_QUEUE_SIZE1; i != (q.rear1 + 1) %
MAX_QUEUE_SIZE1; i = (i + 1) % MAX_QUEUE_SIZE1)
        {
            printf("%d\t", q.array[i]);
        }
    }
}

void display2(Queue q)
{
    if (q.front2 == q.rear2)
    {
        printf("\nThe queue is empty, nothing to print");
    }
    else
    {
        printf("\n");
        for (int i = (q.front2 + 1) % MAX_QUEUE_SIZE; i != (q.rear2 + 1) % MAX_QUEUE_SIZE;
i = (i + 1) % MAX_QUEUE_SIZE)
        {
            printf("%d\t", q.array[i]);
        }
    }
}
```

```
void push1(Queue *q, int key)
{
   if ((q->rear1 + 1) % MAX_QUEUE_SIZE1 == q->front1)
   {
      printf("\nThe queue is full, cannot push");
   }
   else
   {
      q->rear1 = (q->rear1 + 1) % MAX_QUEUE_SIZE1;
      q->array[q->rear1] = key;
   }
}



void push2(Queue *q, int key)
{
   if (((q->rear2 + 1) % MAX_QUEUE_SIZE)+MAX_QUEUE_SIZE2 == q->front2)
   {
      printf("\nThe queue is full, cannot push");
   }
   else
   {
      q->rear2 = (q->rear2 + 1) % MAX_QUEUE_SIZE;
      q->array[q->rear2] = key;
   }
}



int pop1(Queue *q)
{
   int temp = q->array[(q->front1 + 1) % MAX_QUEUE_SIZE1];
   q->front1 = (q->front1 + 1) % MAX_QUEUE_SIZE1;
   return temp;
}



int pop2(Queue *q)
{
   int temp = q->array[(q->front2 + 1) % MAX_QUEUE_SIZE];
   q->front2 = (q->front2 + 1) % MAX_QUEUE_SIZE;
   return temp;
}



int main()
{
   Queue q;
```

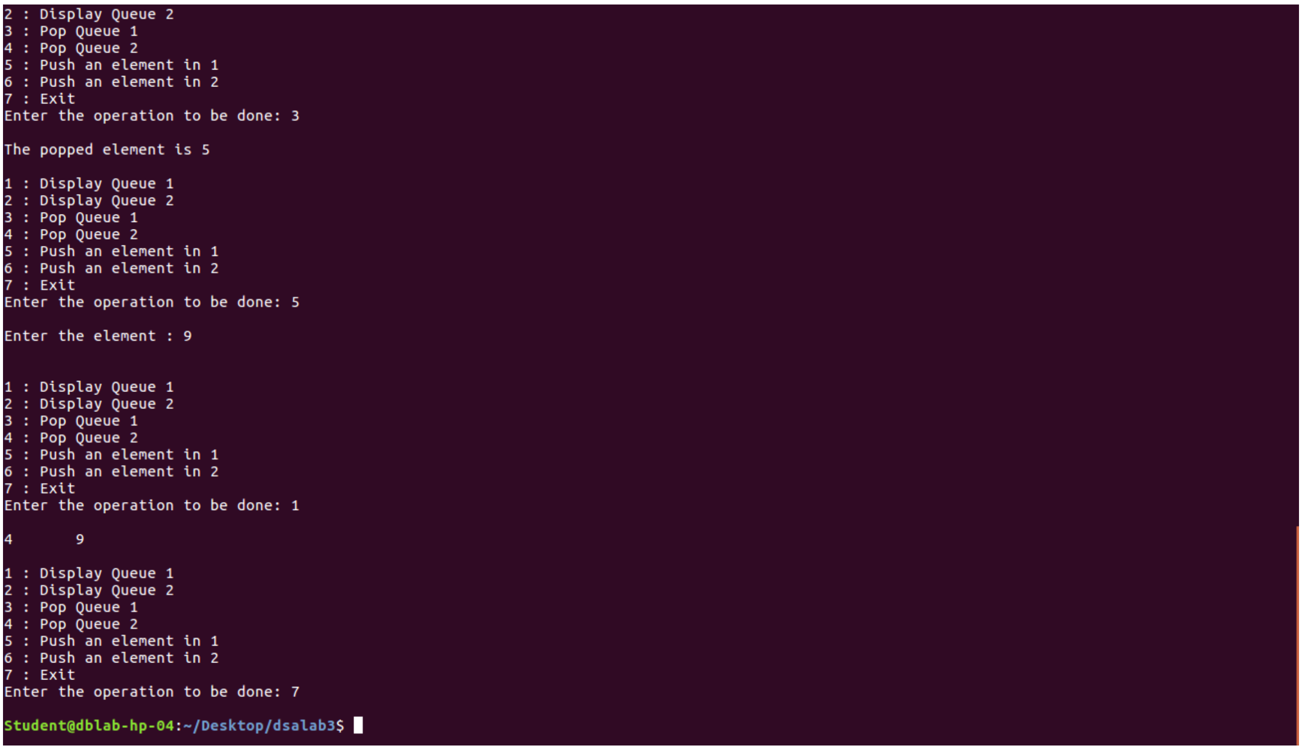
```
    q.front1 = 0;
    q.rear1 = 0;
    q.front2 = MAX_QUEUE_SIZE2;
    q.rear2 = MAX_QUEUE_SIZE2;
    int ch = 0, ele;
    while (ch < 7)
    {
        printf("\n1 : Display Queue 1 \n2 : Display Queue 2 \n3 : Pop Queue 1 \n4 : Pop Queue 2 \n5 : Push an element in 1\n6 : Push an element in 2\n7 : Exit");
        printf("\nEnter the operation to be done: ");
        scanf("%d", &ch);
        switch (ch)
        {
        case 1:
                display1(q);
                break;
        case 2:
                display2(q);
                break;
        case 3:
            if (q.front1 == q.rear1)
            {
                printf("\nThis queue is empty");
            }
            else
            {
                ele = pop1(&q);
                printf("\nThe popped element is %d", ele);
            }
            break;
         case 4:
            if (q.front2 == q.rear2)
            {
                printf("\nThis queue is empty");
            }
            else
            {
                ele = pop2(&q);
                printf("\nThe popped element is %d", ele);
            }
            break;
        case 5:
            printf("\nEnter the element : ");
            scanf("%d", &ele);
            push1(&q, ele);
            break;
        case 6:
            printf("\nEnter the element : ");
            scanf("%d", &ele);
            push2(&q, ele);
            break;
        }
        printf("\n");
    }
    return 0;
}
```



```
Student@dblab-hp-04:~/Desktop/dsalab3$ cc la5q2.c -o la5q2
Student@dblab-hp-04:~/Desktop/dsalab3$ ./la5q2

1 : Display Queue 1
2 : Display Queue 2
3 : Pop Queue 1
4 : Pop Queue 2
5 : Push an element in 1
6 : Push an element in 2
7 : Exit
Enter the operation to be done: 5

Enter the element : 5


1 : Display Queue 1
2 : Display Queue 2
3 : Pop Queue 1
4 : Pop Queue 2
5 : Push an element in 1
6 : Push an element in 2
7 : Exit
Enter the operation to be done: 5

Enter the element : 4


1 : Display Queue 1
2 : Display Queue 2
3 : Pop Queue 1
4 : Pop Queue 2
5 : Push an element in 1
6 : Push an element in 2
7 : Exit
Enter the operation to be done: 1

5       4

1 : Display Queue 1
2 : Display Queue 2
3 : Pop Queue 1
4 : Pop Queue 2
5 : Push an element in 1
```



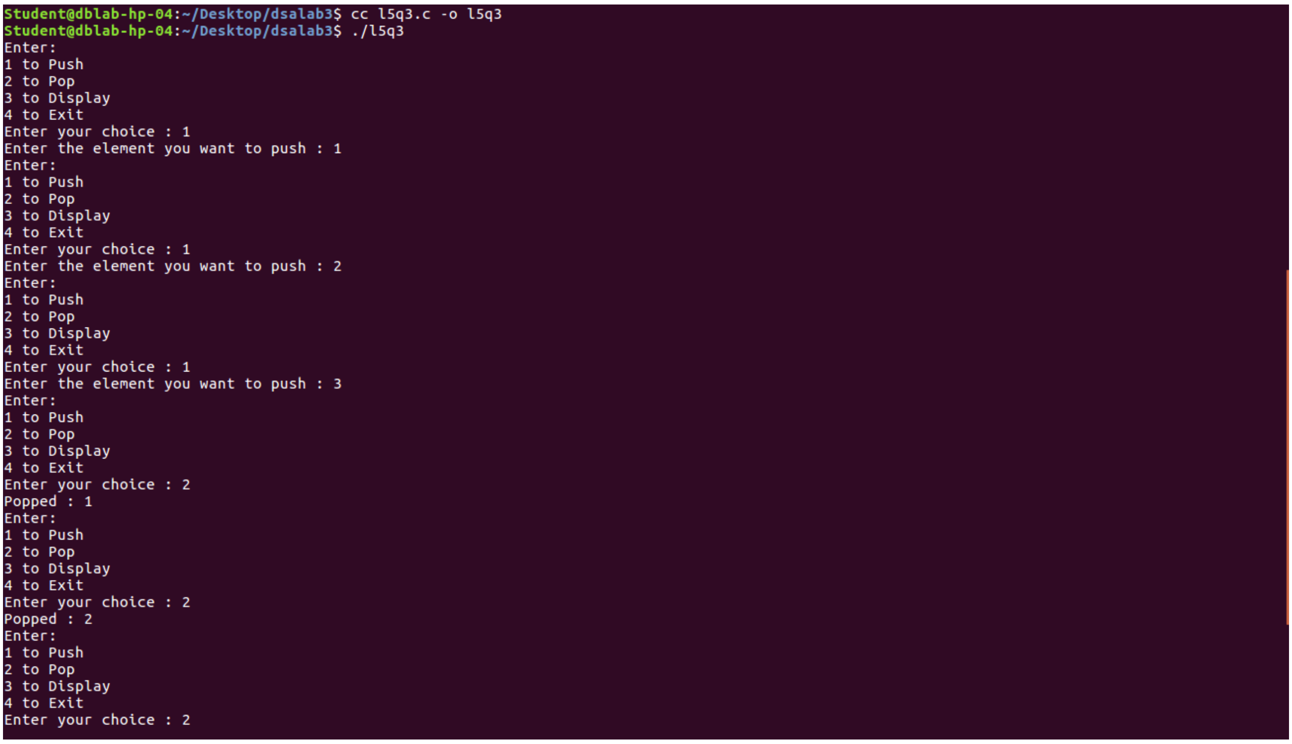
```
Enter the operation to be done: 5

Enter the element : 1

The queue is full, cannot push

1 : Display Queue 1
2 : Display Queue 2
3 : Pop Queue 1
4 : Pop Queue 2
5 : Push an element in 1
6 : Push an element in 2
7 : Exit
Enter the operation to be done: 5

Enter the element : 4

The queue is full, cannot push

1 : Display Queue 1
2 : Display Queue 2
3 : Pop Queue 1
4 : Pop Queue 2
5 : Push an element in 1
6 : Push an element in 2
7 : Exit
Enter the operation to be done: 1

5       4

1 : Display Queue 1
2 : Display Queue 2
3 : Pop Queue 1
4 : Pop Queue 2
5 : Push an element in 1
6 : Push an element in 2
7 : Exit
Enter the operation to be done: 3

The popped element is 5

1 : Display Queue 1
2 : Display Queue 2
```



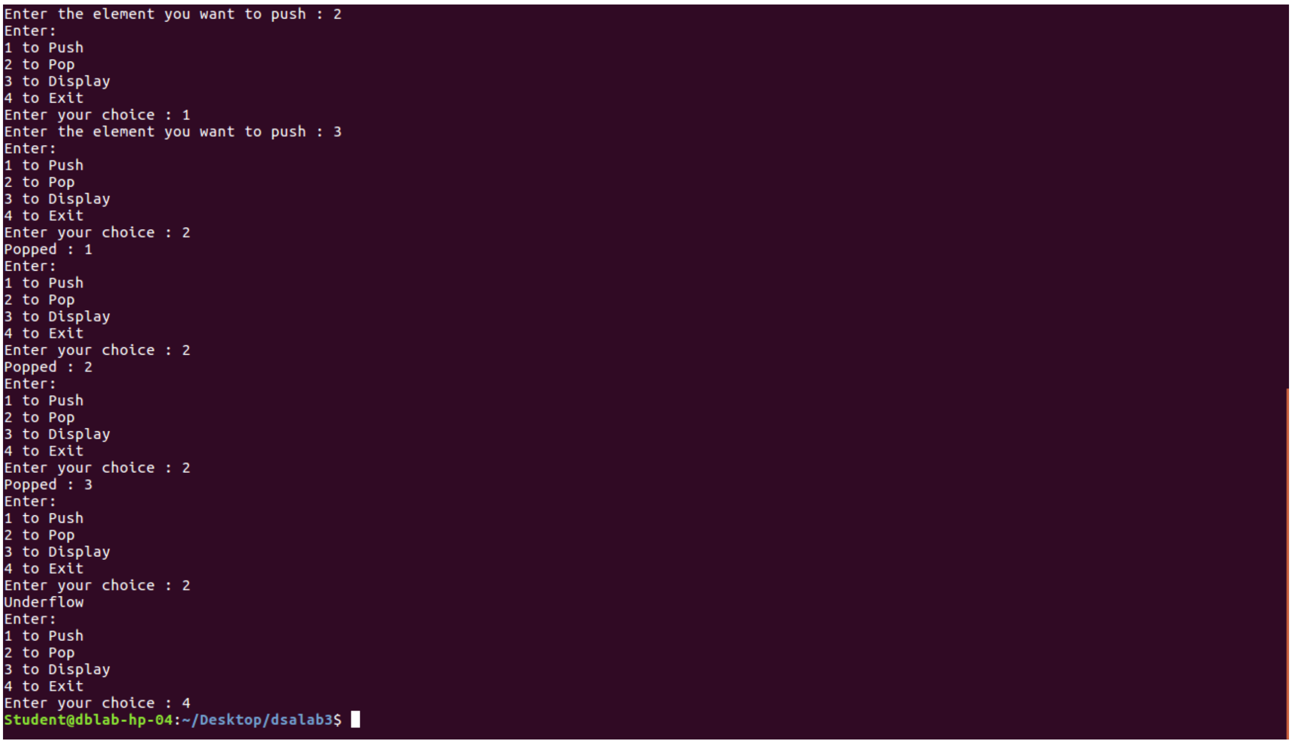
```
2 : Display Queue 2
3 : Pop Queue 1
4 : Pop Queue 2
5 : Push an element in 1
6 : Push an element in 2
7 : Exit
Enter the operation to be done: 3

The popped element is 5

1 : Display Queue 1
2 : Display Queue 2
3 : Pop Queue 1
4 : Pop Queue 2
5 : Push an element in 1
6 : Push an element in 2
7 : Exit
Enter the operation to be done: 5

Enter the element : 9


1 : Display Queue 1
2 : Display Queue 2
3 : Pop Queue 1
4 : Pop Queue 2
5 : Push an element in 1
6 : Push an element in 2
7 : Exit
Enter the operation to be done: 1

4       9

1 : Display Queue 1
2 : Display Queue 2
3 : Pop Queue 1
4 : Pop Queue 2
5 : Push an element in 1
6 : Push an element in 2
7 : Exit
Enter the operation to be done: 7

Student@dblab-hp-04:~/Desktop/dsalab3$ 
```

Q3) Implement a queue with two stacks without transferring the elements of the second stack back to stack one. (use stack1 as an input stack and stack2 as an output stack).

```
#include <stdio.h>
#include <stdlib.h>

#define MAX 5

typedef struct Stack{
        int arr[MAX];
        int top;
}Stack;

int isEmpty(Stack *s) {
        if(s->top==-1)
                return 1;
        return 0;
}

void push(Stack *s,int ch) {
        if((s->top+1)<MAX)
                s->arr[++(s->top)]=ch;
        else
                printf("Overflow!\n");
}

int pop(Stack *s) {
        if(isEmpty(s))
                return -1;
        return s->arr[(s->top)--];
}

int main() {
    Stack s1, s2;
    s1.top = s2.top = -1;
    int ch,n;
    int i=0;
    while (1){
        printf("Enter:\n1 to Push\n2 to Pop\n3 to Display\n4 to Exit\nEnter your choice : ");
        scanf("%d",&ch);
        switch(ch){
            case 1 :
                printf("Enter the element you want to push : ");
                scanf("%d",&n);
                push(&s1,n);
                break;
            case 2 :
                if(isEmpty(&s2)) {
                    while(!isEmpty(&s1)){
                        push(&s2,pop(&s1));
                    }
                    n=pop(&s2);
                    if( n!=-1)
                        printf("Popped : %d\n",n);
                    else
                        printf("Underflow\n");
                }
                else {
                    n=pop(&s2);
                    if(n!=-1)
                        printf("Popped : %d\n",n);
                    else
                        printf("Underflow\n");
                }
                break;
            case 3:
                for(int i=0; i<MAX; i++){
                    printf(" %d", s2.arr[i]);
                }
                printf("\n");
                break;
            case 4:
                exit(0);
        }
    }
    return 0;
}
```



```
Student@dblab-hp-04:~/Desktop/dsalab3$ cc l5q3.c -o l5q3
Student@dblab-hp-04:~/Desktop/dsalab3$ ./l5q3
Enter:
1 to Push
2 to Pop
3 to Display
4 to Exit
Enter your choice : 1
Enter the element you want to push : 1
Enter:
1 to Push
2 to Pop
3 to Display
4 to Exit
Enter your choice : 1
Enter the element you want to push : 2
Enter:
1 to Push
2 to Pop
3 to Display
4 to Exit
Enter your choice : 1
Enter the element you want to push : 3
Enter:
1 to Push
2 to Pop
3 to Display
4 to Exit
Enter your choice : 2
Popped : 1
Enter:
1 to Push
2 to Pop
3 to Display
4 to Exit
Enter your choice : 2
Popped : 2
Enter:
1 to Push
2 to Pop
3 to Display
4 to Exit
Enter your choice : 2
```



```
Enter the element you want to push : 2
Enter:
1 to Push
2 to Pop
3 to Display
4 to Exit
Enter your choice : 1
Enter the element you want to push : 3
Enter:
1 to Push
2 to Pop
3 to Display
4 to Exit
Enter your choice : 2
Popped : 1
Enter:
1 to Push
2 to Pop
3 to Display
4 to Exit
Enter your choice : 2
Popped : 2
Enter:
1 to Push
2 to Pop
3 to Display
4 to Exit
Enter your choice : 2
Popped : 3
Enter:
1 to Push
2 to Pop
3 to Display
4 to Exit
Enter your choice : 2
Underflow
Enter:
1 to Push
2 to Pop
3 to Display
4 to Exit
Enter your choice : 4
Student@dblab-hp-04:~/Desktop/dsalab3$
```