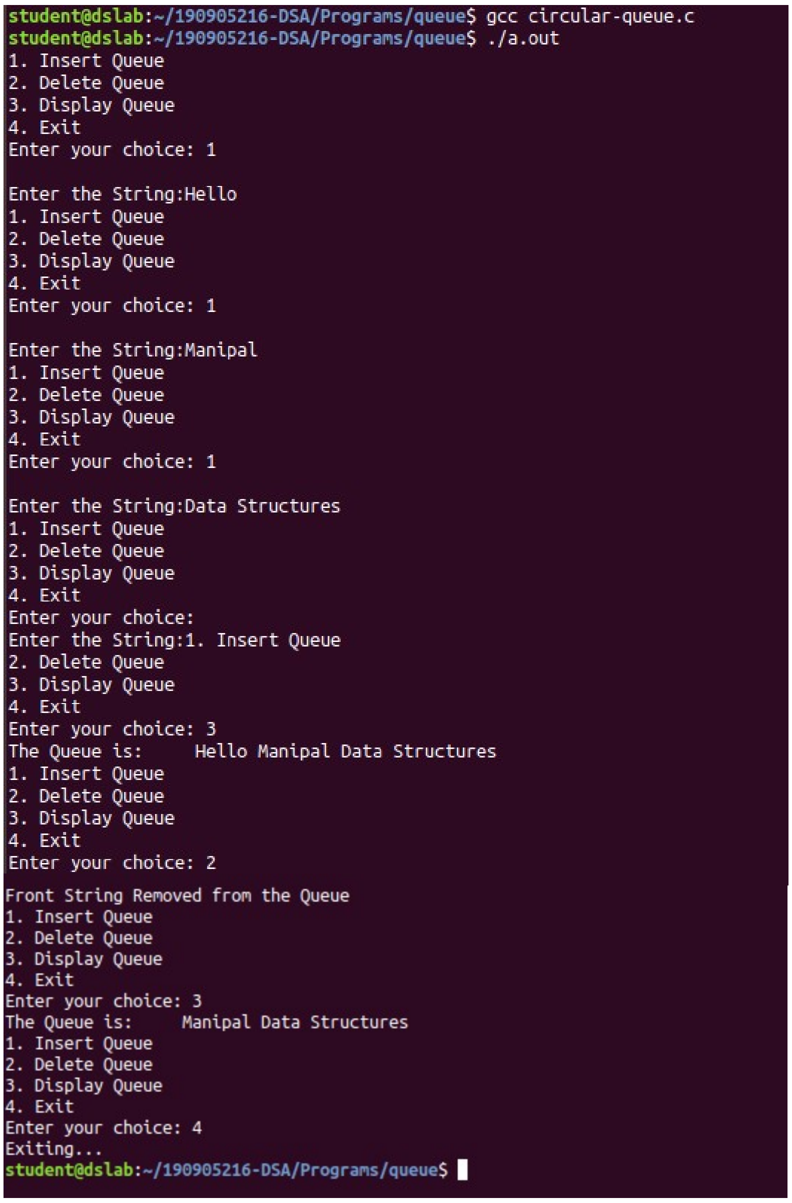
# Queue

Q1:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define MAXSIZE 20 // Max number of Strings
#define MAXSTR 30  // Max Length of strings

typedef struct{
	char **a;
	int front, rear;
}CQ;

void createcq(CQ* cq){
	int i;
	cq->front=-1;
	cq->rear=-1;
	cq->a=malloc(sizeof(char*)*MAXSIZE);
	for(i=0;i<MAXSIZE;i++){
		cq->a[i]=malloc(sizeof(char)*MAXSTR);
	}
}

void insertcq(CQ* cq, char* str){

	if(cq->front==cq->rear && cq->rear==-1){
		cq->front=cq->rear=0;
		strcpy(cq->a[cq->rear],str);
		return;
	}

	if(cq->front == (cq->rear +1)%MAXSIZE){
		printf("Queue is full\n");
		return;
	}

	cq->rear = (cq->rear +1)%MAXSIZE;
	strcpy(cq->a[cq->rear], str);
}

void deletecq(CQ* cq){
	if(cq->front==cq->rear && cq->rear==-1){
		printf("Queue is Empty!!\n");
		return;
	}
	cq->front=(cq->front+1)%MAXSIZE;
	return;
}

void displaycq(CQ* cq){
        int i;
        if(cq->front==cq->rear && cq->rear==-1){
                printf("Queue Empty");
                return;
        }
        printf("The Queue is:     ");
        for(i=cq->front;i<cq->rear;i=(i+1)%MAXSIZE){
                printf("%s ",cq->a[i]);
        }
        printf("%s\n",cq->a[i]);
}

int main(){

        CQ cq;
        createcq(&cq);

        int ch, flag = 1;
        char* x=malloc(sizeof(char) * MAXSTR);
    do
    {
        printf("1. Insert Queue\n2. Delete Queue\n3. Display Queue\n4. Exit\n");
        printf("Enter your choice: ");

        scanf("%d", &ch);
        switch (ch)
        {
        case 1:
            printf("\nEnter the String:");
            scanf("%s", x);
            insertcq(&cq, x);
            break;
        case 2:
            deletecq(&cq);
            printf("\nFront String Removed from the Queue\n");
            break;
        case 3:
            displaycq(&cq);
            break;
        case 4:
            printf("Exiting...\n");
            flag = 0;
            break;
        default:
            printf("\nWrong choice!!! Try Again.\n");
        }
    }while(flag);
    return 0;
}
```

Output:



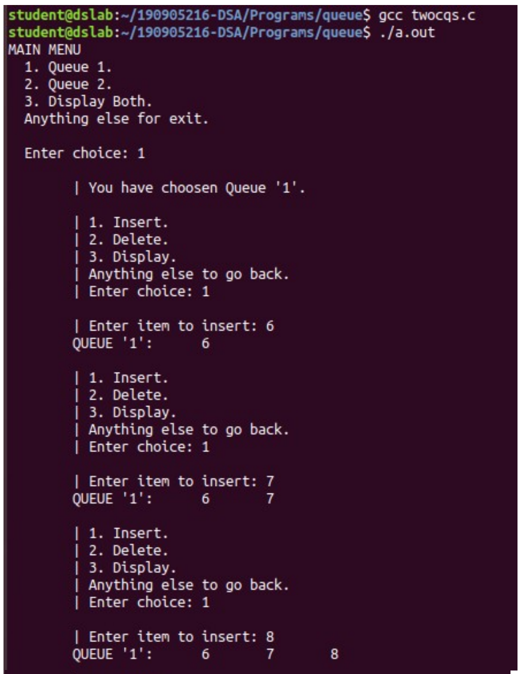
```
student@dslab:~/190905216-DSA/Programs/queue$ gcc circular-queue.c
student@dslab:~/190905216-DSA/Programs/queue$ ./a.out
1. Insert Queue
2. Delete Queue
3. Display Queue
4. Exit
Enter your choice: 1

Enter the String:Hello
1. Insert Queue
2. Delete Queue
3. Display Queue
4. Exit
Enter your choice: 1

Enter the String:Manipal
1. Insert Queue
2. Delete Queue
3. Display Queue
4. Exit
Enter your choice: 1

Enter the String:Data Structures
1. Insert Queue
2. Delete Queue
3. Display Queue
4. Exit
Enter your choice:
Enter the String:1. Insert Queue
2. Delete Queue
3. Display Queue
4. Exit
Enter your choice: 3
The Queue is:     Hello Manipal Data Structures
1. Insert Queue
2. Delete Queue
3. Display Queue
4. Exit
Enter your choice: 2

Front String Removed from the Queue
1. Insert Queue
2. Delete Queue
3. Display Queue
4. Exit
Enter your choice: 3
The Queue is:     Manipal Data Structures
1. Insert Queue
2. Delete Queue
3. Display Queue
4. Exit
Enter your choice: 4
Exiting...
student@dslab:~/190905216-DSA/Programs/queue$
```

Q2:

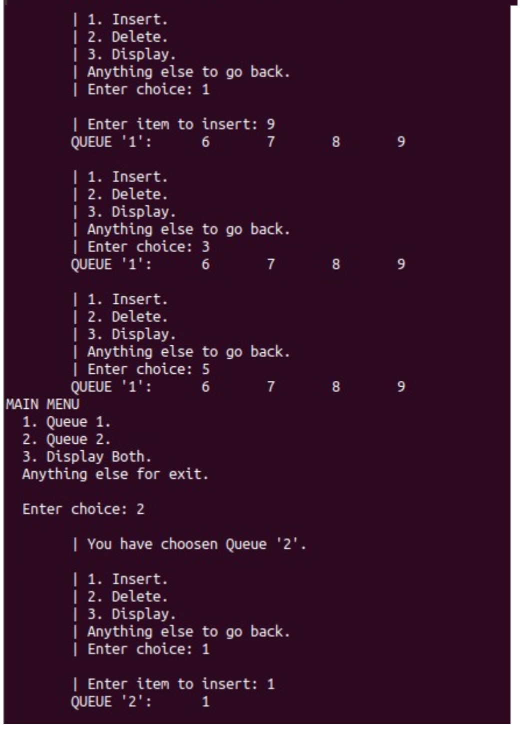
```
#include <stdio.h>
#include <stdlib.h>

#define SIZE 10
#define UNDERFLOW_INT -32767

typedef struct{
   int * arr;
   int front1, rear1, cap1;//front,rear,capacity
   int front2, rear2, cap2;
} CQ;

typedef CQ * CQptr;

typedef enum {
   NO = 0,
   YES = 1,
} BOOL;

//qno is the parameter to decide which q is being called
BOOL isFull (CQ q, int qno) {
   if (qno == 1 && q.cap1 == SIZE/2)
      return YES;
   else if (qno == 2 && q.cap2 == SIZE/2)
      return YES;
   return NO;
}

BOOL isEmpty (CQ q, int qno) {
   if (qno == 1 && q.cap1 == 0)
      return YES;
   else if (qno == 2 && q.cap2 == 0)
      return YES;
   return NO;
}

void insert (CQptr q, int item, int qno) {
   if (isFull(*q, qno)) {
      printf("\nQUEUE '%d' OVERFLOW!", qno);
      return;
   }

   if (qno == 1) {

      if (isEmpty(*q, qno))
         q->front1 = q->rear1 = 0;

      else if (q->rear1 == SIZE/2 - 1)
         q->rear1 = 0;
```

```
        else
            q->rear1 += 1;

        *(q->arr + q->rear1) = item;
        q->cap1++;
    }

    if (qno == 2) {

        if (isEmpty(*q, qno))
            q->front2 = q->rear2 = SIZE - 1;

        else if (q->rear2 == SIZE/2)
            q->rear2 = SIZE - 1;

        else
            q->rear2 -= 1;

        *(q->arr + q->rear2) = item;
        q->cap2++;
    }
}

int delete (CQptr q, int qno) {
    if (isEmpty(*q, qno)) {
        printf("\n\t\tQUEUE '%d' UNDERFLOW!\n\n", qno);
        return UNDERFLOW_INT;
    }

    int item = 0;
    if (qno == 1) {

        item = *(q->arr + q->front1);
        *(q->arr + q->front1) = 0;

        if (q->front1 == q->rear1)
            q->front1 = q->rear1 = -1;

        else if (q->front1 == SIZE/2 - 1)
            q->front1 = 0;

        else
            q->front1 += 1;

        q->cap1--;
    }

    if (qno == 2) {

        item = *(q->arr + q->front2);
        *(q->arr + q->front2) = 0;
        if (q->front2 == q->rear2)
            q->front2 = q->rear2 = SIZE - 1;

        else if (q->front2 == SIZE/2)
            q->front2 = SIZE - 1;

        else
            q->front2 -= 1;

        q->cap2--;
    }
    return item;
}

void display (CQ q, int qno) {
    if (isEmpty(q, qno)) {
        printf("\tEMPTY q %d\n", qno);
        return;
    }

    printf("\tQUEUE '%d': ", qno);
    int i;

    if (qno == 1) {
        if (q.rear1 >= q.front1)
            for (i = q.front1; i <= q.rear1; ++i)
                printf("\t%d", *(q.arr + i));

        else {
            for (i = q.front1; i < SIZE/2; ++i)
                printf("\t%d", *(q.arr + i));
            for (i = 0; i <= q.rear1; ++i)
                printf("\t%d", *(q.arr + i));
        }
    }

    else if (qno == 2) {
        if (q.rear2 <= q.front2)
            for (i = q.front2; i >= q.rear2; --i)
                printf("\t%d", *(q.arr + i));

        else {
            for (i = q.front2; i >= SIZE/2; --i)
                printf("\t%d", *(q.arr + i));
            for (i = SIZE - 1; i >= q.rear2; --i)
                printf("\t%d", *(q.arr + i));
        }
    }

    printf ("\n");
}
```

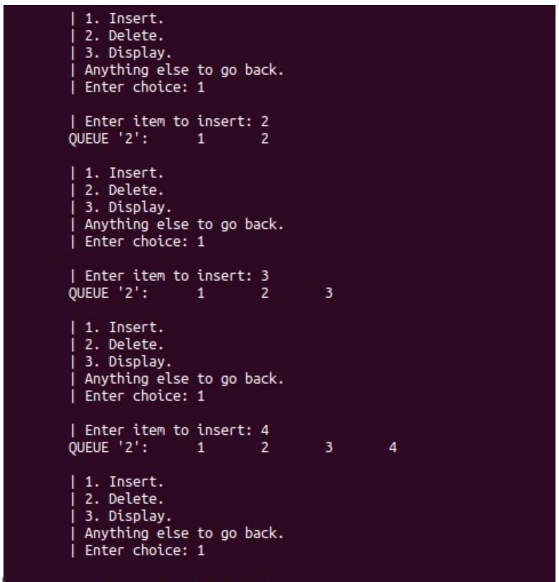
```
int main() {

    CQptr q = (CQptr)malloc(sizeof(CQ));
    q->arr = (int *)calloc(SIZE, sizeof(int));

    q->front1 = q->rear1 = -1;
    q->front2 = q->rear2 = SIZE;
    q->cap1 = q->cap2 = 0;

    int qno;
    do {
        printf("MAIN MENU\n  1. Queue 1.\n  2. Queue 2.\n  3. Display Both.\n  Anything else for 
exit.\n\n  Enter choice: ");
        scanf("%d", &qno);

        if (qno == 3) {
            display(*q, 1);
            display(*q, 2);
            continue;
        }
        else if (!(qno == 1 || qno == 2))
            exit(6);

        printf("\n\t| You have choosen Queue '%d'.\n", qno);
        char choice;

        do {
            printf("\n\t| 1. Insert.\n\t| 2. Delete.\n\t| 3. Display.\n\t| Anything else to go back.\n\t| Enter 
choice: ");
            scanf(" %c", &choice);

            if (choice == '1') {
                int item;
                printf("\n\t| Enter item to insert: ");
                scanf("%d", &item);
                insert(q, item, qno);
            }

            else if (choice == '2') {
                int item = delete(q, qno);
                if (item != UNDERFLOW_INT)
                    printf("\n\t| Deleted Item = %d.\n", item);
            }

            display(*q, qno);

        } while (choice == '1' || choice == '2' || choice == '3');

    } while (qno == 1 || qno == 2 || qno == 3);

}
```



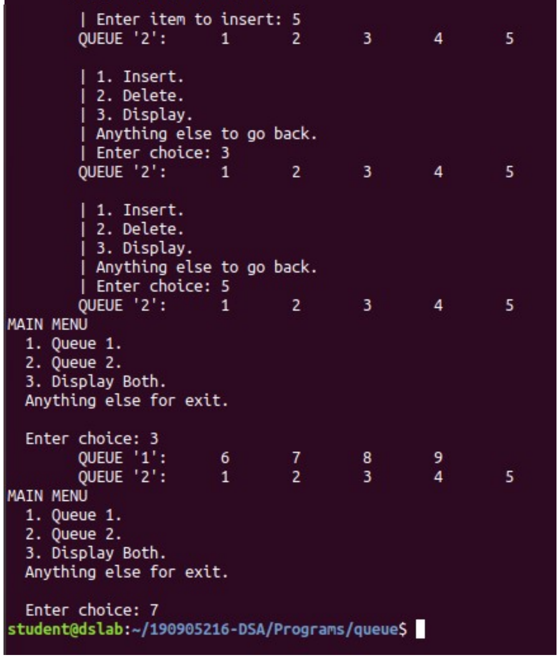
```
student@dslab:~/190905216-DSA/Programs/queue$ gcc twocqs.c
student@dslab:~/190905216-DSA/Programs/queue$ ./a.out
MAIN MENU
  1. Queue 1.
  2. Queue 2.
  3. Display Both.
  Anything else for exit.

  Enter choice: 1

        | You have choosen Queue '1'.

        | 1. Insert.
        | 2. Delete.
        | 3. Display.
        | Anything else to go back.
        | Enter choice: 1

        | Enter item to insert: 6
        QUEUE '1':      6

        | 1. Insert.
        | 2. Delete.
        | 3. Display.
        | Anything else to go back.
        | Enter choice: 1

        | Enter item to insert: 7
        QUEUE '1':      6       7

        | 1. Insert.
        | 2. Delete.
        | 3. Display.
        | Anything else to go back.
        | Enter choice: 1

        | Enter item to insert: 8
        QUEUE '1':      6       7       8
```



```
        | 1. Insert.
        | 2. Delete.
        | 3. Display.
        | Anything else to go back.
        | Enter choice: 1

        | Enter item to insert: 9
        QUEUE '1':      6       7       8       9

        | 1. Insert.
        | 2. Delete.
        | 3. Display.
        | Anything else to go back.
        | Enter choice: 3
        QUEUE '1':      6       7       8       9

        | 1. Insert.
        | 2. Delete.
        | 3. Display.
        | Anything else to go back.
        | Enter choice: 5
        QUEUE '1':      6       7       8       9
MAIN MENU
  1. Queue 1.
  2. Queue 2.
  3. Display Both.
  Anything else for exit.

  Enter choice: 2

        | You have choosen Queue '2'.

        | 1. Insert.
        | 2. Delete.
        | 3. Display.
        | Anything else to go back.
        | Enter choice: 1

        | Enter item to insert: 1
        QUEUE '2':      1
```



```
        | 1. Insert.
        | 2. Delete.
        | 3. Display.
        | Anything else to go back.
        | Enter choice: 1

        | Enter item to insert: 2
        QUEUE '2':      1       2

        | 1. Insert.
        | 2. Delete.
        | 3. Display.
        | Anything else to go back.
        | Enter choice: 1

        | Enter item to insert: 3
        QUEUE '2':      1       2       3

        | 1. Insert.
        | 2. Delete.
        | 3. Display.
        | Anything else to go back.
        | Enter choice: 1

        | Enter item to insert: 4
        QUEUE '2':      1       2       3       4

        | 1. Insert.
        | 2. Delete.
        | 3. Display.
        | Anything else to go back.
        | Enter choice: 1
```



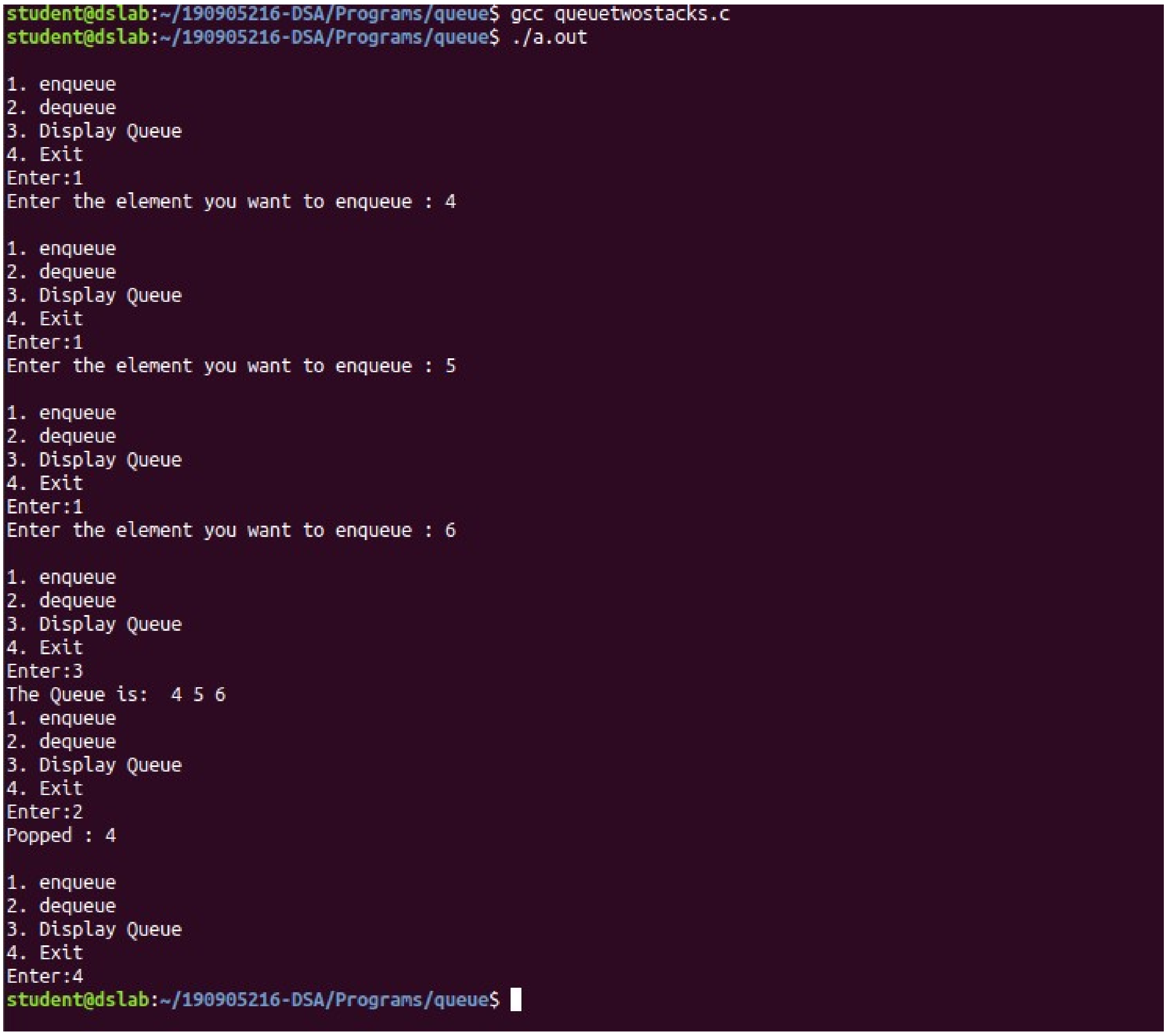
```
        | Enter item to insert: 5
        QUEUE '2':      1       2       3       4       5

        | 1. Insert.
        | 2. Delete.
        | 3. Display.
        | Anything else to go back.
        | Enter choice: 3
        QUEUE '2':      1       2       3       4       5

        | 1. Insert.
        | 2. Delete.
        | 3. Display.
        | Anything else to go back.
        | Enter choice: 5
        QUEUE '2':      1       2       3       4       5
MAIN MENU
  1. Queue 1.
  2. Queue 2.
  3. Display Both.
  Anything else for exit.

  Enter choice: 3
        QUEUE '1':      6       7       8       9
        QUEUE '2':      1       2       3       4       5
MAIN MENU
  1. Queue 1.
  2. Queue 2.
  3. Display Both.
  Anything else for exit.

  Enter choice: 7
student@dslab:~/190905216-DSA/Programs/queue$
```

**Q3:**

```
#include<stdio.h>
#include<stdlib.h>
#define MAXSIZE 5
typedef struct {
        int arr[MAXSIZE];
         int top;
}Stack;

int isEmpty(Stack *s) {
 if(s->top==-1)
        return 1;
 return 0;
}
void enqueue(Stack *s,int ch) {
        if((s->top+1)<MAXSIZE)
                s->arr[++(s->top)]=ch;
        else printf("Overflow!\n");
}
int dequeue(Stack *s) {
 if(isEmpty(s))
        return -9999;
 return s->arr[(s->top)--];
}

void display(Stack *s){
        if(s->top==-1){
                printf("Queue Empty");
        }
        else{
                printf("The Queue is:  ");
                for(int i=0;i<=s->top;i++){
                        printf("%d ",s->arr[i]);
                }
        }
}
int main() {
        Stack s1, s2;
        s1.top=s2.top=-1;
        int ch,n; int i=0;
        while (1){
           printf("\n1. enqueue\n2. dequeue\n3. Display Queue\n4. Exit\nEnter:");
           scanf("%d",&ch);
           switch(ch){
                case 1 :
                        printf("Enter the element you want to enqueue : ");
                        scanf("%d",&n);
                        enqueue(&s1,n);
                        break;
                case 2 :
                        if(isEmpty(&s2)) {
```

```
                    while(!isEmpty(&s1)){
                        enqueue(&s2,dequeue(&s1));
                    }
                    n=dequeue(&s2);
                    if( n!=-9999)
                        printf("Popped : %d\n",n);
                    else
                        printf("Underflow\n");
                }
                else{
                    n=dequeue(&s2);
                    if(n!=-9999)
                        printf("Popped : %d\n",n);
                    else
                        printf("Underflow\n");
                }
                break;
            case 3:
                display(&s1);
                break;
            case 4:
                exit(0);
        }
    }
    return 0;
}
```

PTO----->



```
student@dslab:~/190905216-DSA/Programs/queue$ gcc queuetwostacks.c
student@dslab:~/190905216-DSA/Programs/queue$ ./a.out

1. enqueue
2. dequeue
3. Display Queue
4. Exit
Enter:1
Enter the element you want to enqueue : 4

1. enqueue
2. dequeue
3. Display Queue
4. Exit
Enter:1
Enter the element you want to enqueue : 5

1. enqueue
2. dequeue
3. Display Queue
4. Exit
Enter:1
Enter the element you want to enqueue : 6

1. enqueue
2. dequeue
3. Display Queue
4. Exit
Enter:3
The Queue is:  4 5 6
1. enqueue
2. dequeue
3. Display Queue
4. Exit
Enter:2
Popped : 4

1. enqueue
2. dequeue
3. Display Queue
4. Exit
Enter:4
student@dslab:~/190905216-DSA/Programs/queue$ 
```