Lab 5

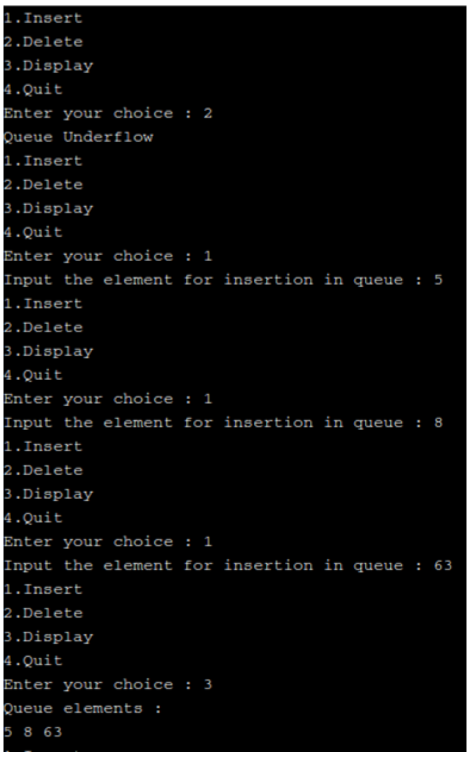
```
//queue_fun.h
#include<stdio.h>
# define MAX 5

int cqueue_arr[MAX];
int front = -1;
int rear = -1;

void insertcq(int item)
{
   if((front == 0 && rear == MAX-1) || (front == rear+1))
   {
      printf("Queue Overflow\n");
      return;
   }
   if(front == -1)
   {
      front = 0;
      rear = 0;
   }
   else
   {
      if(rear == MAX-1)
         rear = 0;
      else
      rear = rear+1;
   }
   cqueue_arr[rear] = item ;
}
void deletecq()
{
   if(front == -1)
   {
   printf("Queue Underflow\n");
   return ;
   }
   printf("Element deleted from queue is : %d\n",cqueue_arr[front]);
   if(front == rear)
   {
      front = -1;
      rear=-1;
   }
   else
   {
      if(front == MAX-1)
         front = 0;
      else
         front = front+1;
   }
}
```

```
void displaycq()
{
   int front_pos = front,rear_pos = rear;
   if(front == -1)
   {
      printf("Queue is empty\n");
      return;
   }
   printf("Queue elements :\n");
   if( front_pos <= rear_pos )
      while(front_pos <= rear_pos)
      {
         printf("%d ",cqueue_arr[front_pos]);
         front_pos++;
      }
   else
   {
      while(front_pos <= MAX-1)
      {
         printf("%d ",cqueue_arr[front_pos]);
         front_pos++;
      }
      front_pos = 0;
      while(front_pos <= rear_pos)
      {
         printf("%d ",cqueue_arr[front_pos]);
         front_pos++;
      }
   }
   printf("\n");
}
```

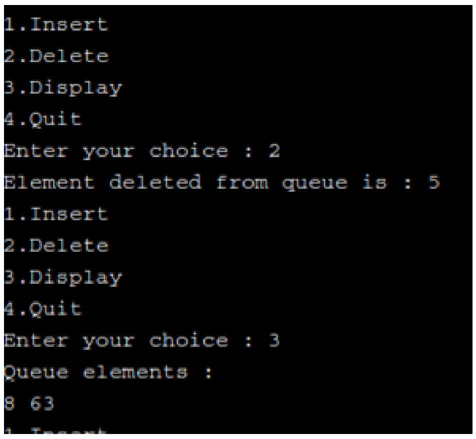
Q1

```
#include <stdio.h>
#include "queue_fun.h"

int main()
{
   int choice,item;
   do
   {
      printf("1.Insert\n");
      printf("2.Delete\n");
      printf("3.Display\n");
      printf("4.Quit\n");
      printf("Enter your choice : ");
      scanf("%d",&choice);
      switch(choice)
      {
      case 1 :
         printf("Input the element for insertion in queue : ");
         scanf("%d", &item);
```

```c
            insertcq(item);
            break;
        case 2 :
            deletecq();
            break;
        case 3:
            displaycq();
            break;
        case 4:
            break;
        default:
            printf("\nWrong choice!!! Try Again.\n");
        }
    }while(choice!=4);
    return 0;
}
```



```
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 2
Queue Underflow
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 1
Input the element for insertion in queue : 5
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 1
Input the element for insertion in queue : 8
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 1
Input the element for insertion in queue : 63
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 3
Queue elements :
5 8 63
```



```
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 2
Element deleted from queue is : 5
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 3
Queue elements :
8 63
```

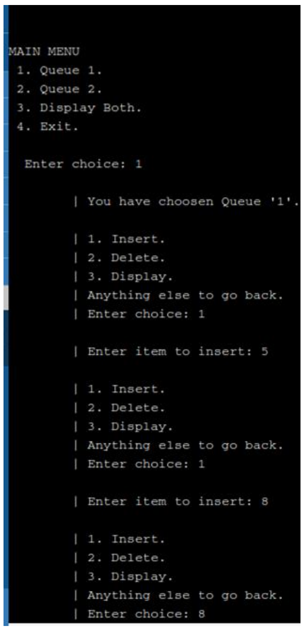
Q2

```c
#include <stdio.h>
#include <stdlib.h>
#define SIZE 10
#define UNDERFLOW_INT -32767
// Boolean type, just for readability
typedef enum
{
    NO = 0,
    YES = 1,
} BOOL;
```

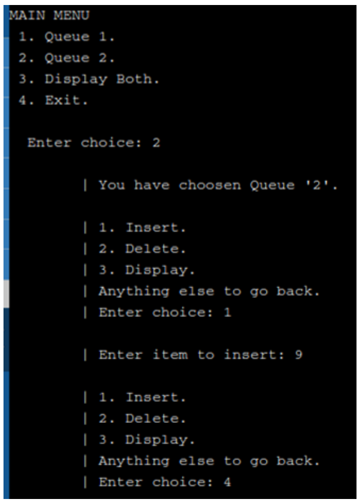
```
typedef struct CircularQueue
{
   int * arr;
   int front1, rear1, cap1;
   int front2, rear2, cap2;
} CQUEUE_t;

typedef CQUEUE_t * CQUEUE_p_t;

// Queue methods

BOOL isFullQueue (CQUEUE_t queue, int qno)
{
   if (qno == 1 && queue.cap1 == SIZE/2)
      return YES;
   else if (qno == 2 && queue.cap2 == SIZE/2)
      return YES;
   return NO;
}

BOOL isEmptyQueue (CQUEUE_t queue, int qno)
{
   if (qno == 1 && queue.cap1 == 0)
   return YES;
   else if (qno == 2 && queue.cap2 == 0)
   return YES;
   return NO;
}

void insert (CQUEUE_p_t queue, int item, int qno)
{
   if (isFullQueue(*queue, qno))
   {
      printf("\n\t\tQUEUE '%d' OVERFLOW!\n\n", qno);
      return;
   }

   if (qno == 1)
   {
      if (isEmptyQueue(*queue, qno))
         queue->front1 = queue->rear1 = 0;
      else if (queue->rear1 == SIZE/2 - 1)
         queue->rear1 = 0;
      else
         queue->rear1 += 1;
      *(queue->arr + queue->rear1) = item;
      queue->cap1++;
   }
   if (qno == 2)
   {
      if (isEmptyQueue(*queue, qno))
         queue->front2 = queue->rear2 = SIZE - 1;
```

```
        else if (queue->rear2 == SIZE/2)
            queue->rear2 = SIZE - 1;
        else
            queue->rear2 -= 1;
        *(queue->arr + queue->rear2) = item;
        queue->cap2++;
    }
}

int delete (CQUEUE_p_t queue, int qno)
{
    if (isEmptyQueue(*queue, qno))
    {
        printf("\n\t\tQUEUE '%d' UNDERFLOW!\n\n", qno);
        return UNDERFLOW_INT;
    }
    int item = 0;
    if (qno == 1)
    {
        item = *(queue->arr + queue->front1);
        *(queue->arr + queue->front1) = 0;
        if (queue->front1 == queue->rear1)
            queue->front1 = queue->rear1 = -1;
        else if (queue->front1 == SIZE/2 - 1)
            queue->front1 = 0;
        else
            queue->front1 += 1;
        queue->cap1--;
    }
    if (qno == 2)
    {
        item = *(queue->arr + queue->front2);
        *(queue->arr + queue->front2) = 0;
        if (queue->front2 == queue->rear2)
            queue->front2 = queue->rear2 = SIZE - 1;
        else if (queue->front2 == SIZE/2)
            queue->front2 = SIZE - 1;
        else
            queue->front2 -= 1;
        queue->cap2--;
    }
    return item;
}

void display (CQUEUE_t queue, int qno)
{
    if (isEmptyQueue(queue, qno))
    {
        printf("\n\t\tEMPTY QUEUE %d.\n\n", qno);
        return;
    }
    printf("\n\tQUEUE '%d': ", qno);
    int i;
    if (qno == 1)
    {
        if (queue.rear1 >= queue.front1)
            for (i = queue.front1; i <= queue.rear1; ++i)
                printf("\t%d", *(queue.arr + i));
        else
        {
            for (i = queue.front1; i < SIZE/2; ++i)
                printf("\t%d", *(queue.arr + i));
            for (i = 0; i <= queue.rear1; ++i)
                printf("\t%d", *(queue.arr + i));
        }
    }
    else if (qno == 2)
    {
        if (queue.rear2 <= queue.front2)
            for (i = queue.front2; i >= queue.rear2; --i)
                printf("\t%d", *(queue.arr + i));
        else
        {
            for (i = queue.front2; i >= SIZE/2; --i)
                printf("\t%d", *(queue.arr + i));
            for (i = SIZE - 1; i >= queue.rear2; --i)
                printf("\t%d", *(queue.arr + i));
        }
    }
    printf ("\n\n");
}

int main(int argc, const char * argv[])
{
    //printf("\n\n  Two circular queues in a single array.\n  The initial SIZE = 10\n  Initially, for queue
1, front and rear are set to -1, and for queue 2 to SIZE.\n\n");
    CQUEUE_p_t queue = (CQUEUE_p_t)malloc(sizeof(CQUEUE_t));
    queue->arr = (int *)calloc(SIZE, sizeof(int));
    queue->front1 = queue->rear1 = -1;
    queue->front2 = queue->rear2 = SIZE;
    queue->cap1 = queue->cap2 = 0;
    int item;
    int qno;
    do{
        printf("\n\nMAIN MENU\n 1. Queue 1.\n 2. Queue 2.\n 3. Display Both.\n 4. Exit.\n\n  Enter
choice: ");
        scanf("%d", &qno);
        if (qno == 3)
        {
            display(*queue, 1);
            display(*queue, 2);
            continue;
        }
        else if (!(qno == 1 || qno == 2))
```

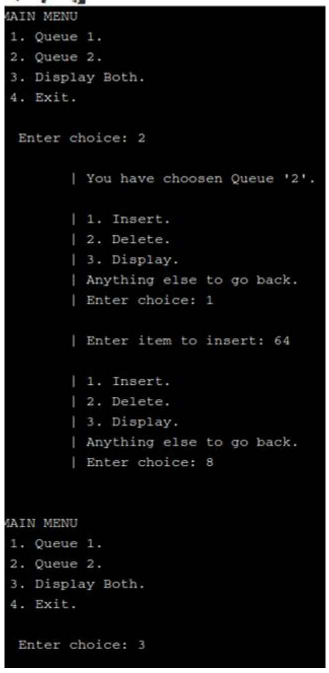
```
            exit(6);
        printf("\n\t| You have choosen Queue '%d'.\n", qno);
        int ch;
        do {
            printf("\n\t| 1. Insert.\n\t| 2. Delete.\n\t| 3. Display.\n\t| Anything else to go back.\n\t| Enter 
choice: ");
            scanf(" %d", &ch);
            switch(ch)
            {
                case 1:
                    printf("\n\t| Enter item to insert: ");
                    scanf("%d", &item);
                    insert(queue, item, qno);
                    break;
                case 2:
                    item = delete(queue, qno);
                    if (item != UNDERFLOW_INT)
                        printf("\n\t| Deleted Item = %d.\n", item);
                    break;
                case 3:
                    display(*queue, qno);
            }
        } while (ch<4);
    } while (qno!=4);
    return 0;
}
```



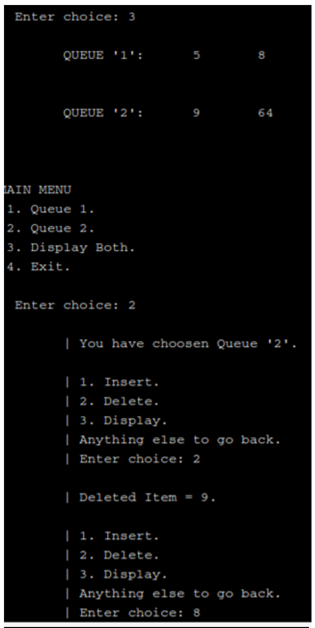
```
MAIN MENU
 1. Queue 1.
 2. Queue 2.
 3. Display Both.
 4. Exit.

  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: 5

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

        | Enter item to insert: 8

        | 1. Insert.
        | 2. Delete.
        | 3. Display.
        | Anything else to go back.
        | Enter choice: 8
```



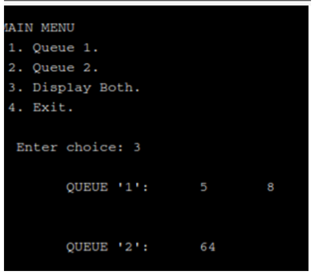
```
MAIN MENU
 1. Queue 1.
 2. Queue 2.
 3. Display Both.
 4. Exit.

  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: 9

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



```
MAIN MENU
 1. Queue 1.
 2. Queue 2.
 3. Display Both.
 4. 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: 64

        | 1. Insert.
        | 2. Delete.
        | 3. Display.
        | Anything else to go back.
        | Enter choice: 8


MAIN MENU
 1. Queue 1.
 2. Queue 2.
 3. Display Both.
 4. Exit.

  Enter choice: 3
```



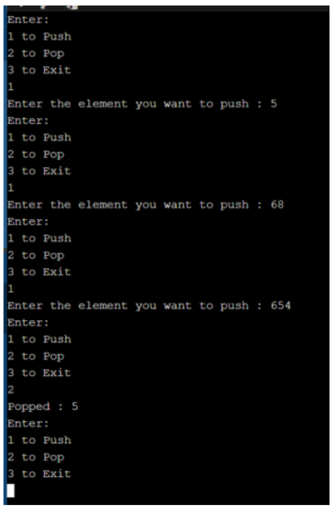
```
  Enter choice: 3

        QUEUE '1':      5       8


        QUEUE '2':      9       64



MAIN MENU
 1. Queue 1.
 2. Queue 2.
 3. Display Both.
 4. Exit.

  Enter choice: 2

        | You have choosen Queue '2'.

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

        | Deleted Item = 9.

        | 1. Insert.
        | 2. Delete.
        | 3. Display.
        | Anything else to go back.
        | Enter choice: 8
```



```
MAIN MENU
 1. Queue 1.
 2. Queue 2.
 3. Display Both.
 4. Exit.

  Enter choice: 3

        QUEUE '1':      5       8


        QUEUE '2':      64
```

Q3

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 5

typedef struct
{
   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 -9999;
   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 Exit\n");
      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!=-9999)
                        printf("Popped : %d\n",n);
                    else
                        printf("Underflow\n");
                }
                else
                {
                    n=pop(&s2);
                    if(n!=-9999)
                        printf("Popped : %d\n",n);
                    else
                        printf("Underflow\n");
                }
                break;
            case 3:
                exit(0);
        }
    }
    return 0;
}
```



```
Enter:
1 to Push
2 to Pop
3 to Exit
1
Enter the element you want to push : 5
Enter:
1 to Push
2 to Pop
3 to Exit
1
Enter the element you want to push : 68
Enter:
1 to Push
2 to Pop
3 to Exit
1
Enter the element you want to push : 654
Enter:
1 to Push
2 to Pop
3 to Exit
2
Popped : 5
Enter:
1 to Push
2 to Pop
3 to Exit
```