DS LAB

Lab 5

Q1)

Source Code:
"cqueue.h"

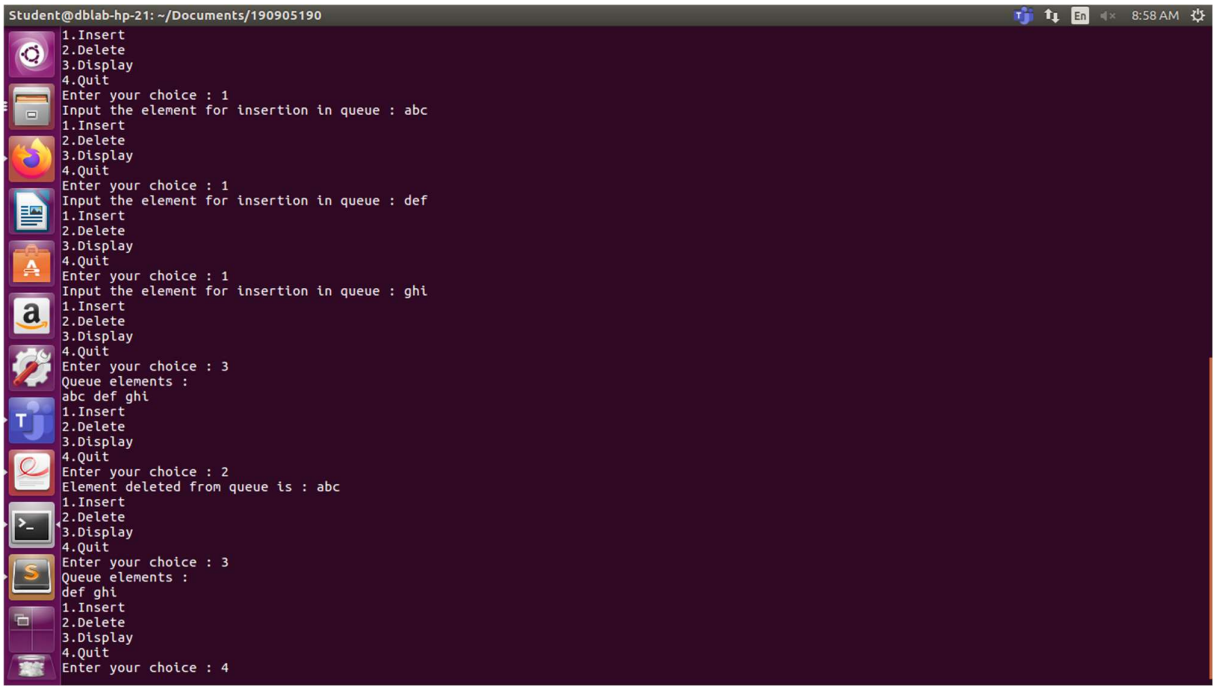
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX 5
#define MAXL 100

typedef struct
{
	char items[MAX][MAXL];
	int front, rear;
} CQUEUE;

void init(CQUEUE * cq)
{
	cq->front = -1;
	cq->rear = -1;
}

void insertcq(CQUEUE * cq, char str[])
{
	if((cq->front == 0 && cq->rear == MAX - 1) || (cq->front == cq->rear + 1))
	{
		printf("Queue Overflow\n");
		return;
	}
	if(cq->front == -1)
	{
		cq->front = 0;
		cq->rear = 0;
	}
	else
	{
		if(cq->rear == MAX -1)
			cq->rear = 0;
		else
			cq->rear = cq->rear + 1;
	}
	strcpy(cq->items[cq->rear], str);
}

void deletecq(CQUEUE * cq)
{
	if(cq->front == -1)
```

```
        {
                printf("Queue Underflow\n");
                return;
        }
        printf("Element deleted from queue is : %s\n", cq->items[cq->front]);
    if (cq->front == cq->rear)
    {
        cq->front = -1;
        cq->rear = -1;
    }
    else
    {
        if (cq->front == MAX - 1)
            cq->front = 0;
        else
            cq->front = cq->front + 1;
    }
}

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

“q1.c”

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

int main()
{
   int ch;
   char item[MAXL];
   CQUEUE * cq;
   cq = malloc(sizeof(CQUEUE));
   init(cq);
   do
   {
      printf("1.Insert\n");
      printf("2.Delete\n");
      printf("3.Display\n");
      printf("4.Quit\n");
      printf("Enter your choice : ");
      scanf("%d", &ch);
      switch (ch)
      {
      case 1:
         printf("Input the element for insertion in queue : ");
         scanf("%s", item);
         insertcq(cq, item);
         break;
      case 2:
         deletecq(cq);
         break;
      case 3:
         displaycq(cq);
         break;
      case 4:
         break;
      default:
         printf("\nWrong choice!!! Try Again.\n");
      }
   } while (ch != 4);
   return 0;
}
```

Output:



Q2)

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 10
#define UNDERFLOW_INT -32767

typedef struct CircularQueue
{
   int *arr;
   int front1, rear1, cap1;
   int front2, rear2, cap2;
} CQUEUE;
typedef CQUEUE *CQUEUEptr;
// Queue methods
int isFullQueue(CQUEUE cq, int q_no)
{
   if (q_no == 1 && cq.cap1 == SIZE / 2)
      return 1;
   else if (q_no == 2 && cq.cap2 == SIZE / 2)
      return 1;
   return 0;
}
int isEmptyQueue(CQUEUE cq, int q_no)
{
   if (q_no == 1 && cq.cap1 == 0)
      return 1;
   else if (q_no == 2 && cq.cap2 == 0)
      return 1;
```

```
    return 0;
}
void insert(CQUEUEptr cq, int item, int q_no)
{
    if (isFullQueue(*cq, q_no))
    {
        printf("\n\t\tQUEUE '%d' OVERFLOW!\n\n", q_no);
        return;
    }
    if (q_no == 1)
    {
        if (isEmptyQueue(*cq, q_no))
            cq->front1 = cq->rear1 = 0;
        else if (cq->rear1 == SIZE / 2 - 1)
            cq->rear1 = 0;
        else
            cq->rear1 += 1;
        *(cq->arr + cq->rear1) = item;
        cq->cap1++;
    }
    if (q_no == 2)
    {
        if (isEmptyQueue(*cq, q_no))
            cq->front2 = cq->rear2 = SIZE - 1;
        else if (cq->rear2 == SIZE / 2)
            cq->rear2 = SIZE - 1;
        else
            cq->rear2 -= 1;
        *(cq->arr + cq->rear2) = item;
        cq->cap2++;
    }
}
int delete (CQUEUEptr cq, int q_no)
{
    if (isEmptyQueue(*cq, q_no))
    {
        printf("\n\t\tQUEUE '%d' UNDERFLOW!\n\n", q_no);
        return UNDERFLOW_INT;
    }
    int item = 0;
    if (q_no == 1)
    {
        item = *(cq->arr + cq->front1);
        *(cq->arr + cq->front1) = 0;
        if (cq->front1 == cq->rear1)
            cq->front1 = cq->rear1 = -1;
        else if (cq->front1 == SIZE / 2 - 1)
            cq->front1 = 0;
        else
            cq->front1 += 1;
        cq->cap1--;
    }
```

```
    if (q_no == 2)
    {
        item = *(cq->arr + cq->front2);
        *(cq->arr + cq->front2) = 0;
        if (cq->front2 == cq->rear2)
            cq->front2 = cq->rear2 = SIZE - 1;
        else if (cq->front2 == SIZE / 2)
            cq->front2 = SIZE - 1;
        else
            cq->front2 -= 1;
        cq->cap2--;
    }
    return item;
}
void display(CQUEUE cq, int q_no)
{
    if (isEmptyQueue(cq, q_no))
    {
        printf("\n\t\tEMPTY QUEUE %d.\n\n", q_no);
        return;
    }
    printf("\n\tQUEUE '%d': ", q_no);
    int i;
    if (q_no == 1)
    {
        if (cq.rear1 >= cq.front1)
            for (i = cq.front1; i <= cq.rear1; ++i)
                printf("\t%d", *(cq.arr + i));
        else
        {
            for (i = cq.front1; i < SIZE / 2; ++i)
                printf("\t%d", *(cq.arr + i));
            for (i = 0; i <= cq.rear1; ++i)
                printf("\t%d", *(cq.arr + i));
        }
    }
    else if (q_no == 2)
    {
        if (cq.rear2 <= cq.front2)
            for (i = cq.front2; i >= cq.rear2; --i)
                printf("\t%d", *(cq.arr + i));
        else
        {
            for (i = cq.front2; i >= SIZE / 2; --i)
                printf("\t%d", *(cq.arr + i));
            for (i = SIZE - 1; i >= cq.rear2; --i)
                printf("\t%d", *(cq.arr + i));
        }
    }
    printf("\n\n");
}
```

```
int main()
{
   CQUEUEptr cq = (CQUEUEptr)malloc(sizeof(CQUEUE));
   cq->arr = (int *)calloc(SIZE, sizeof(int));
   cq->front1 = cq->rear1 = -1;
   cq->front2 = cq->rear2 = SIZE;
   cq->cap1 = cq->cap2 = 0;
   int item;
   int q_no;
   do
   {
      printf("MAIN MENU\n 1. Queue 1.\t 2. Queue 2.\t 3. Display Both.\t 4. Exit.\n Enter choice:
");
      scanf("%d", &q_no);
      if (q_no == 3)
      {
         display(*cq, 1);
         display(*cq, 2);
         continue;
      }
      else if (!(q_no == 1 || q_no == 2))
         exit(6);
      printf("\tQueue '%d'.\n", q_no);
      int ch;
      do
      {
         printf("\t1. Insert.\t 2. Delete.\t 3. Display.\t| Anything else to go back.\n\tEnter choice: ");
         scanf(" %d", &ch);
         switch (ch)
         {
         case 1:
            printf("\tEnter item to insert: ");
            scanf("%d", &item);
            insert(cq, item, q_no);
            break;
         case 2:
            item = delete (cq, q_no);
            if (item != UNDERFLOW_INT)
               printf("\n\t| Deleted Item = %d.\n", item);
            break;
         case 3:
            display(*cq, q_no);
         }
      } while (ch < 4);
   } while (q_no != 4);
   return 0;
}
```

Ouput:

```
Student@dblab-hp-21: ~/Documents/190905190                                  9:17 AM
Student@dblab-hp-21:~/Documents/190905190$ ./q2
MAIN MENU
 1. Queue 1.     2. Queue 2.     3. Display Both.        4. Exit.
 Enter choice: 1
        Queue '1'.
        1. Insert.       2. Delete.      3. Display.    | Anything else to go back.
        Enter choice: 1
        Enter item to insert: 2
        1. Insert.       2. Delete.      3. Display.    | Anything else to go back.
        Enter choice: 1
        Enter item to insert: 3
        1. Insert.       2. Delete.      3. Display.    | Anything else to go back.
        Enter choice: 1
        Enter item to insert: 4
        1. Insert.       2. Delete.      3. Display.    | Anything else to go back.
        Enter choice: 3

        QUEUE '1':      2       3       4

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

        QUEUE '2':      1       2       3

        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.
```

```
Student@dblab-hp-21: ~/Documents/190905190                                  9:18 AM
        1. Insert.       2. Delete.      3. Display.    | Anything else to go back.
        Enter choice: 1
        Enter item to insert: 4
        1. Insert.       2. Delete.      3. Display.    | Anything else to go back.
        Enter choice: 3

        QUEUE '1':      2       3       4

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

        QUEUE '2':      1       2       3

        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: 3

        QUEUE '1':      2       3       4


        QUEUE '2':      1       2       3

MAIN MENU
 1. Queue 1.     2. Queue 2.     3. Display Both.        4. Exit.
 Enter choice: 4
Student@dblab-hp-21:~/Documents/190905190$
```

Q3)

Source Code:

```
#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)--];
}
void display(STACK *s)
{
   if(isEmpty(s))
      printf("Stack is Empty\n");
   else
   {
      for (int i = 0; i <= s->top; i++)
      {
         printf("%d\t", s->arr[i]);
      }
   }
}

int main()
{
   STACK * s1, * s2;
   s1 = malloc(sizeof(STACK));
   s2 = malloc(sizeof(STACK));
   s1->top = s2->top = -1;
   int ch, n;
   int i = 0;
```

```
    printf("Enter:\n1 to Push\n2 to Pop\n3 to Display\n4 to Exit\n");
    while (1)
    {
        printf("Enter 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 != -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:
            printf("Stack 1: \t");
            display(s1);
            printf("\n");
            printf("Stack 2: \t");
            display(s2);
            break;
        case 4:
            exit(0);
        }
    }
    return 0;
}
```

Output:

```
Student@dblab-hp-21: ~/Documents/190905190
Student@dblab-hp-21:~/Documents/190905190$ gcc -c q3.c
Student@dblab-hp-21:~/Documents/190905190$ gcc -o q3 q3.o
Student@dblab-hp-21:~/Documents/190905190$ ./q3
Enter:
1 to Push
2 to Pop
3 to Display
4 to Exit
Enter Choice: 1
Enter the element you want to push : 1
Enter Choice: 1
Enter the element you want to push : 2
Enter Choice: 1
Enter the element you want to push : 3
Enter Choice: 1
Enter the element you want to push : 4
Enter Choice: 1
Enter the element you want to push : 5
Enter Choice: 3
Stack 1:        1       2       3       4       5
Stack 2:        Stack is Empty
Enter Choice: 2
Popped : 1
Enter Choice: 2
Popped : 2
Enter Choice: 2
Popped : 3
Enter Choice: 2
Popped : 4
Enter Choice: 3
Stack 1:        Stack is Empty

Stack 2:        5       Enter Choice: 4
Student@dblab-hp-21:~/Documents/190905190$
```