LAB 5

Q1

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

#define MAX 10

typedef struct {
    char q[MAX][MAX];
    int f, r, count;
} cq;


void cq_init(cq* c) {
    c->f = 0;
    c->r = -1;
    c->count = 0;
}


bool cq_full(cq* c) {
    if (c->count == MAX) {
        return true;
    }

    else return false;
}


bool cq_empty(cq* c) {
    if (c->count == 0) {
        return true;
    }
    else return false;
}


void insertcq(cq* c, char* n) {
    if (cq_full(c)) {
        printf("Queue is full\n");
        return;
    }

    c->r = (c->r + 1) % MAX;
    strcpy(c->q[c->r], n);
    c->count++;
}


char* deletecq(cq* c) {
```

```
    if (cq_empty(c)) {
        printf("Queue is empty\n");
        return 0;
    }

    int temp = c->f;
    c->f = (c->f+1) % MAX;
    c->count--;
    return c->q[temp];
}


void displaycq(cq* c) {
    if (cq_empty(c)) {
        printf("Queue is empty\n");
        return;
    }

    for (int i = c->f; i != c->r; i = (i + 1) % MAX) {
        printf("%s\n", c->q[i]);
    }

    printf("%s\n", c->q[c->r]);
}


int main() {
    cq cque;
    cq* c = &cque;
    cq_init(c);

    int choice;
    char* 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 : ");
                scanf("%s", item);
                insertcq(c, item);
                break;
            case 2 :
                deletecq(c);
                break;
            case 3:
```

```
            displaycq(c);
            break;
        case 4:
            break;
        default:
            printf("\nInvalid option\n");
      }
   }while(choice!=4);

   return 0;
}
```

```
student@V310Z-000:~/Desktop/rhea/dsa$ cc circularq.c -o circularq
student@V310Z-000:~/Desktop/rhea/dsa$ ./circularq
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 1
Input the element for insertion : College
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 1
Input the element for insertion : is
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 1
Input the element for insertion : MIT
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 3
College
is
MIT
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 2
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 3
is
MIT
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 4
```

Q2

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

typedef struct {
    int* q;
    int size;
    int f1, f2;
    int r1, r2;
    int count1, count2;
} cqd;

void cqd_init(cqd* c, int n) {
    c->q = (int*)calloc(n, sizeof(int));
    c->size = n;
    c->r1 = -1;
    c->f1 = -1;
    c->r2 = (n / 2);
    c->f2 = (n / 2);
    c->count1 = 0;
    c->count2 = 0;
}


bool cqd_empty(cqd* c, int qno) {
    if (qno == 1) {
        return !(c->count1);
    }
    else if (qno == 2) {
        return !(c->count2);
    }
}


bool cqd_full(cqd* c, int qno) {
    if (qno == 1)
    {
        return c->count1 == (c->size / 2) ? true : false;
    }
    else if (qno == 2)
    {
        return c->count2 == (c->size / 2) ? true : false;
    }
}


void insertcqd(cqd* c, int qno, int e) {
```

```
    if (qno == 1) {
        if (cqd_full(c, qno)) {
            printf("Queue 1 is full");
            return;
        }

        c->r1 = (c->r1 + 1) % (c->size / 2);
        c->q[c->r1] = e;
        if(cqd_empty(c, qno)){
            c->f1=c->r1;
        }
        c->count1++;
        return;
    }
    else if (qno == 2) {
        if (cqd_full(c, qno))
        {
            printf("Queue 2 is full");
            return;
        }

        c->r2 = (c->r2 + 1) % (c->size / 2) + c->size / 2;
        c->q[c->r2] = e;
        if(cqd_empty(c, qno)){
            c->f2=c->r2;
        }
        c->count2++;
        return;
    }
}


int deletecqd(cqd* c, int qno) {
    if (qno == 1) {
        if (cqd_empty(c, qno)) {
            printf("Queue 1 is empty\n");
            return 0;
        }

        int element = c->q[c->f1];
        c->f1 = (c->f1 + 1) % (c->size / 2);
        c->count1--;
        return element;
    }
    else if (qno == 2) {
        if (cqd_empty(c, qno))
        {
            printf("Queue 2 is empty\n");
            return 0;
        }

        int element = c->q[c->f2];
```

```
        c->f2 = (c->f2 + 1) % (c->size / 2) + (c->size / 2);
        c->count2--;
        return element;
    }
}


void display(cqd* c, int qno) {
    if (qno == 1) {
        if (cqd_empty(c, qno)) {
            printf("Queue 1 is empty");
            return;
        }

        for (int i = c->f1; i != c->r1; i = (i + 1) % (c->size / 2))
        {
            printf("%d, ", c->q[i]);
        }

        printf("%d\n", c->q[c->r1]);
    }
    else if (qno == 2) {
        if (cqd_empty(c, qno))
        {
            printf("Queue 2 is empty \n");
            return;
        }

        for (int i = c->f2; i != c->r2; i = (i + 1) % (c->size / 2) + (c->size / 2))
        {
            printf("%d, ", c->q[i]);
        }

        printf("%d\n", c->q[c->r2]);
    }
}


int main() {
    cqd cqdouble;
    cqd* c = &cqdouble;

    cqd_init(c, 10);

    int n, ch, qno;

    printf("\n1 - Insert");
    printf("\n2 - Delete");
    printf("\n3 - Display");
    printf("\n4 - Exit");

    while (1)
```

```
    {
        printf("\nEnter queue number and choice: ");
        scanf("%d %d", &qno, &ch);

        switch (ch)
        {
        case 1:
            printf("\nEnter value to be inserted : ");
            scanf("%d",&n);
            insertcqd(c, qno, n);
            break;
        case 2:
            deletecqd(c, qno);
            break;
        case 3:
            display(c, qno);
            break;
        case 4:
            exit(0);
        default:
            printf("\nInvalid option\n");
        }
    }

    return 0;
}
```

```
student@V310Z-000:~/Desktop/rhea/dsa$ cc circq2.c -o circq2
student@V310Z-000:~/Desktop/rhea/dsa$ ./circq2

1 - Insert
2 - Delete
3 - Display
4 - Exit
Enter queue number and choice: 1
34

Invalid option

Enter queue number and choice: 1 1

Enter value to be inserted : 23

Enter queue number and choice: 2 1

Enter value to be inserted : 34

Enter queue number and choice: 1 1

Enter value to be inserted : 4

Enter queue number and choice: 1 1

Enter value to be inserted : 6

Enter queue number and choice: 1 2

Enter queue number and choice: 1 3
4, 6

Enter queue number and choice: 2 3
34

Enter queue number and choice: 4
```

Q3 IMPLEMENTING 2 STACKS

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
   int data;
   struct node *next;
};
void push(struct node** top, int data);
int pop(struct node** top);
struct queue
{
   struct node *stack1;
   struct node *stack2;
};
void enqueue(struct queue *q, int x)
{
   push(&q->stack1, x);
}
void dequeue(struct queue *q)
{
   int x;
   if (q->stack1 == NULL && q->stack2 == NULL) {
      printf("queue is empty");
      return;
   }
   if (q->stack2 == NULL) {
      while (q->stack1 != NULL) {
      x = pop(&q->stack1);
      push(&q->stack2, x);
      }
   }
   x = pop(&q->stack2);
   printf("Deleted: %d\n", x);
}
void push(struct node** top, int data)
{
   struct node* newnode = (struct node*) malloc(sizeof(struct node));
      if (newnode == NULL) {
         printf("Stack overflow \n");
         return;
      }
   newnode->data = data;
   newnode->next = (*top);
   (*top) = newnode;
}
int pop(struct node** top)
{
   int buff;
```

```
    struct node *t;
    if (*top == NULL) {
        printf("Stack underflow \n");
    }
    else {
        t = *top;
        buff = t->data;
        *top = t->next;
        free(t);
        return buff;
    }
}
void display(struct node *top1,struct node *top2)
{
    while (top1 != NULL) {
        printf("%d ", top1->data);
        top1 = top1->next;
    }
    while (top2 != NULL) {
        printf("%d ", top2->data);
        top2 = top2->next;
    }
    printf("\n");
}
int main()
{
    struct queue *q = (struct queue*)malloc(sizeof(struct queue));
    int f = 0, a;
    char ch = 'y';
    q->stack1 = NULL;
    q->stack2 = NULL;
    while (ch == 'y'||ch == 'Y') {
        printf("enter ur choice\n1.insert\n2.delete\n3.display\n4.exit\nEnter: ");
        scanf("%d", &f);
        switch(f) {
            case 1 : printf("enter the element to be added to queue: ");
                     scanf("%d", &a);
                     enqueue(q, a);
                     break;
            case 2 : dequeue(q);
                     break;
            case 3 : display(q->stack1, q->stack2);
                     break;
            case 4 : exit(1);
                     break;
            default : printf("invalid\n");
                      break;
        }
    }
}
```

```
student@V310Z-000:~$ cd Desktop/rhea/dsa
student@V310Z-000:~/Desktop/rhea/dsa$ cc stack2.c
student@V310Z-000:~/Desktop/rhea/dsa$ ./stack2
enter ur choice
1.insert
2.delete
3.display
4.exit
Enter: 1
enter the element to be added to queue: 10
enter ur choice
1.insert
2.delete
3.display
4.exit
Enter: 1
enter the element to be added to queue: 5
enter ur choice
1.insert
2.delete
3.display
4.exit
Enter: 2
Deleted: 10
enter ur choice
1.insert
2.delete
3.display
4.exit
Enter: 3
5
enter ur choice
1.insert
2.delete
3.display
4.exit
Enter: 1
enter the element to be added to queue: 6
```

```
enter ur choice
1.insert
2.delete
3.display
4.exit
Enter: 1
enter the element to be added to queue: 7
enter ur choice
1.insert
2.delete
3.display
4.exit
Enter: 1
enter the element to be added to queue: 9
enter ur choice
1.insert
2.delete
3.display
4.exit
Enter: 3
9 7 6 5
enter ur choice
1.insert
2.delete
3.display
4.exit
Enter: 2
Deleted: 5
enter ur choice
1.insert
2.delete
3.display
4.exit
Enter:
```