Lab – 7

Stack And Queue Using Linked List

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

struct Node {
    int data;
    struct Node *link;
};

typedef struct Node node;

//Stack
node *top = NULL;

void push();
void pop();
void displayStack();

void push() {
    node *new1 = (node*)malloc(sizeof(node));
    if (new1 == NULL) {
        printf("\nStack Overflow.\n");
        return;
    }

    printf("\nEnter Value to Push: ");
    scanf("%d", &new1->data);
    new1->link = top;
    top = new1;
}

void pop() {
    if (top == NULL) {
        printf("\nStack Underflow.\n");
        return;
    }

    node *temp = top;
    printf("\nPopped Element: %d\n", temp->data);
    top = top->link;
    free(temp);
}
```

```c
void displayStack() {
    if (top == NULL) {
        printf("\nThe Stack is Empty.\n");
        return;
    }

    printf("\nElements in the Stack: ");
    node *temp = top;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->link;
    }
    printf("\n");
}


//Queue
node *front = NULL, *rear = NULL;

void insert();
void del();
void displayQueue();

void insert() {
    node *new1 = (node*)malloc(sizeof(node));
    if (new1 == NULL) {
        printf("\nQueue Full.\n");
        return;
    }

    printf("\nEnter Value to Insert: ");
    scanf("%d", &new1->data);
    new1->link = NULL;

    if (rear == NULL) {
        front = rear = new1;
        return;
    }
    rear->link = new1;
    rear = new1;
}

void del() {
    if (front == NULL) {
        printf("\nQueue Empty.\n");
        return;
    }

    node *temp = front;
```

```
    printf("\nDeleted Element: %d\n", temp->data);
    front = front->link;

    if (front == NULL) {
        rear = NULL;
    }
    free(temp);
}

void displayQueue() {
    if (front == NULL) {
        printf("\nThe Queue is Empty.\n");
        return;
    }

    printf("\nElements in the Queue: ");
    node *temp = front;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->link;
    }
    printf("\n");
}


// Main
void main(){
    int ch;

    while (1) {
        printf("\n1. Push (Stack) \n2. Pop (Stack) \n3. Display (Stack)");
        printf("\n4. Insert (Queue) \n5. Delete (Queue) \n6. Display (Queue)
\n7. Exit");
        printf("\nEnter Your Choice: ");
        scanf("%d", &ch);

        switch (ch) {
            case 1:
                push();
                break;
            case 2:
                pop();
                break;
            case 3:
                displayStack();
                break;
            case 4:
                insert();
                break;
```

```
            case 5:
                del();
                break;
            case 6:
                displayQueue();
                break;
            case 7:
                exit(0);
            default:
                printf("\nEnter Your Choice: \n");
        }
    }
}
```

Output:

```
1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 1

Enter Value to Push: 10

1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 1

Enter Value to Push: 20

1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 1

Enter Value to Push: 30

1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 3

Elements in the Stack: 30 20 10

1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 2

Popped Element: 30
```

```
1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 2

Popped Element: 20

1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 2

Popped Element: 10

1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 2

Stack Underflow.

1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 4

Enter Value to Insert: 11

1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 4
```

```
Enter Your Choice: 4

Enter Value to Insert: 22

1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 4

Enter Value to Insert: 33

1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 6

Elements in the Queue: 11 22 33

1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 5

Deleted Element: 11

1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 5

Deleted Element: 22
```

```
1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 5

Deleted Element: 22

1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 5

Deleted Element: 33

1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 5

Queue Empty.

1. Push (Stack)
2. Pop (Stack)
3. Display (Stack)
4. Insert (Queue)
5. Delete (Queue)
6. Display (Queue)
7. Exit
Enter Your Choice: 7
PS C:\Users\STUDENT\Desktop\1BM23CS016>
```