

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct node {
5     int data;
6     struct node *next;
7 };
8
9 struct node *top = NULL;
10 void push(int value) {
11     struct node *newNode = (struct node*)malloc(sizeof(struct node));
12     newNode->data = value;
13     newNode->next = top;
14     top = newNode;
15     printf("Pushed: %d\n", value);
16 }
17
18 void pop() {
19     if (top == NULL) {
20         printf("Stack Underflow\n");
21         return;
22     }
23     struct node *temp = top;
24     printf("Popped: %d\n", temp->data);
25     top = top->next;
26     free(temp);
27 }
28 void peekStack() {
29     if (top == NULL) {
30         printf("Stack is empty\n");
31         return;
32     }
33     printf("Top = %d\n", top->data);
34 }
35
36 void peekStack() {
37     if (top == NULL) {
38         printf("Stack is empty\n");
39         return;
40     }
41     printf("Top = %d\n", top->data);
42 }
43 void displayStack() {
44     if (top == NULL) {
45         printf("Stack is empty\n");
46         return;
47     }
48     struct node *temp = top;
49     printf("Stack: ");
50     while (temp != NULL) {
51         printf(" %d ", temp->data);
52         temp = temp->next;
53     }
54     printf("\n");
55 }
56 struct node *front = NULL;
57 struct node *rear = NULL;
58 void enqueue(int value) {
59     struct node *newNode = (struct node*)malloc(sizeof(struct node));
60     newNode->data = value;
61     newNode->next = NULL;
62
63     if (rear == NULL) {
64         front = rear = newNode;
65     } else {
66         rear->next = newNode;
67         rear = newNode;
68     }
69
70     printf("Enqueued: %d\n", value);
71 }

```

```
1 void enqueue(int value) {
2
3     void dequeue() {
4         if (front == NULL) {
5             printf("Queue Underflow\n");
6             return;
7         }
8
9         struct node *temp = front;
10        printf("Dequeued: %d\n", temp->data);
11
12        front = front->next;
13        if (front == NULL)
14            rear = NULL;
15
16        free(temp);
17    }
18
19    void peekQueue() {
20        if (front == NULL) {
21            printf("Queue is empty\n");
22            return;
23        }
24        printf("Front = %d\n", front->data);
25    }
26
27
28    void displayQueue() {
29        if (front == NULL) {
30            printf("Queue is empty\n");
31            return;
32        }
33
34        struct node *temp = front;
35        printf("Queue: ");
36        while (temp != NULL) {
37            printf("%d ", temp->data);
38            temp = temp->next;
39        }
40        printf("\n");
41    }
42}
```

```
    printf("\n");
}int main() {

    printf("\n--- STACK OPERATIONS ---\n");
    push(10);
    push(20);
    push(30);
    displayStack();
    pop();
    displayStack();
    peekStack();

    printf("\n--- QUEUE OPERATIONS ---\n");
    enqueue(5);
    enqueue(15);
    enqueue(25);
    displayQueue();
    dequeue();
    displayQueue();
    peekQueue();

    return 0;
}
```

```
--- STACK OPERATIONS ---
Pushed: 10
Pushed: 20
Pushed: 30
Stack: 30 20 10
Popped: 30
Stack: 20 10
Top = 20

--- QUEUE OPERATIONS ---
Enqueued: 5
Enqueued: 15
Enqueued: 25
Queue: 5 15 25
Dequeued: 5
Queue: 15 25
Front = 15
PS C:\Users\Admin\Desktop\dslab\output>
```