LAB-6a (Sort,Reverse,Concat-SLL)

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

struct node {
    int data;
    struct node *next;
};

struct node *head1 = NULL;
struct node *head2 = NULL;

struct node* create(struct node *head) {
    int n, i, val;
    struct node *temp, *newnode;
    printf("Enter number of nodes: ");
    scanf("%d", &n);
    for (i = 0; i < n; i++) {
        newnode = (struct node *)malloc(sizeof(struct node));
        printf("Enter data: ");
        scanf("%d", &val);
        newnode->data = val;
        newnode->next = NULL;
        if (head == NULL) {
            head = newnode;
            temp = head;
        } else {
            temp->next = newnode;
            temp = newnode;
        }
    }
    return head;
}

void display(struct node *head) {
    struct node *temp;
    if (head == NULL) {
        printf("List is empty\n");
        return;
    }
    temp = head;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

void sort(struct node *head) {
    struct node *i, *j;
    int temp;
    if (head == NULL)
        return;
    for (i = head; i->next != NULL; i = i->next) {
        for (j = i->next; j != NULL; j = j->next) {
            if (i->data > j->data) {
                temp = i->data;
                i->data = j->data;
                j->data = temp;
            }
        }
    }
}

struct node* reverse(struct node *head) {
    struct node *prev = NULL, *curr = head, *next;
    while (curr != NULL) {
        next = curr->next;
        curr->next = prev;
        prev = curr;
        curr = next;
    }
    return prev;
}

struct node* concatenate(struct node *head1, struct node *head2) {
    struct node *temp;
    if (head1 == NULL)
        return head2;
    temp = head1;
    while (temp->next != NULL)
```

OUTPUT:

```
C:\Users\Hrishikesh\OneDrive\Desktop
1.Create List 1
2.Create List 2
3.Sort List 1
4.Reverse List 1
5.Concatenate
6.Display List 1
7.Display List 2
8.Exit
Enter choice: 1
Enter number of nodes: 2
Enter data: 10
Enter data: 20

1.Create List 1
2.Create List 2
3.Sort List 1
4.Reverse List 1
5.Concatenate
6.Display List 1
7.Display List 2
8.Exit
Enter choice: 2
Enter number of nodes: 3
Enter data: 40
Enter data: 50
Enter data: 60

1.Create List 1
2.Create List 2
3.Sort List 1
4.Reverse List 1
5.Concatenate
6.Display List 1
7.Display List 2
8.Exit
Enter choice: 3

1.Create List 1
2.Create List 2
3.Sort List 1
4.Reverse List 1
5.Concatenate
6.Display List 1
7.Display List 2
8.Exit
Enter choice: 6
10 20
```

```
1.Create List 1
2.Create List 2
3.Sort List 1
4.Reverse List 1
5.Concatenate
6.Display List 1
7.Display List 2
8.Exit
Enter choice: 6
10 20

1.Create List 1
2.Create List 2
3.Sort List 1
4.Reverse List 1
5.Concatenate
6.Display List 1
7.Display List 2
8.Exit
Enter choice: 5

1.Create List 1
2.Create List 2
3.Sort List 1
4.Reverse List 1
5.Concatenate
6.Display List 1
7.Display List 2
8.Exit
Enter choice: 6
10 20 40 50 60

1.Create List 1
2.Create List 2
3.Sort List 1
4.Reverse List 1
5.Concatenate
6.Display List 1
7.Display List 2
8.Exit
Enter choice: 4

1.Create List 1
2.Create List 2
3.Sort List 1
4.Reverse List 1
```

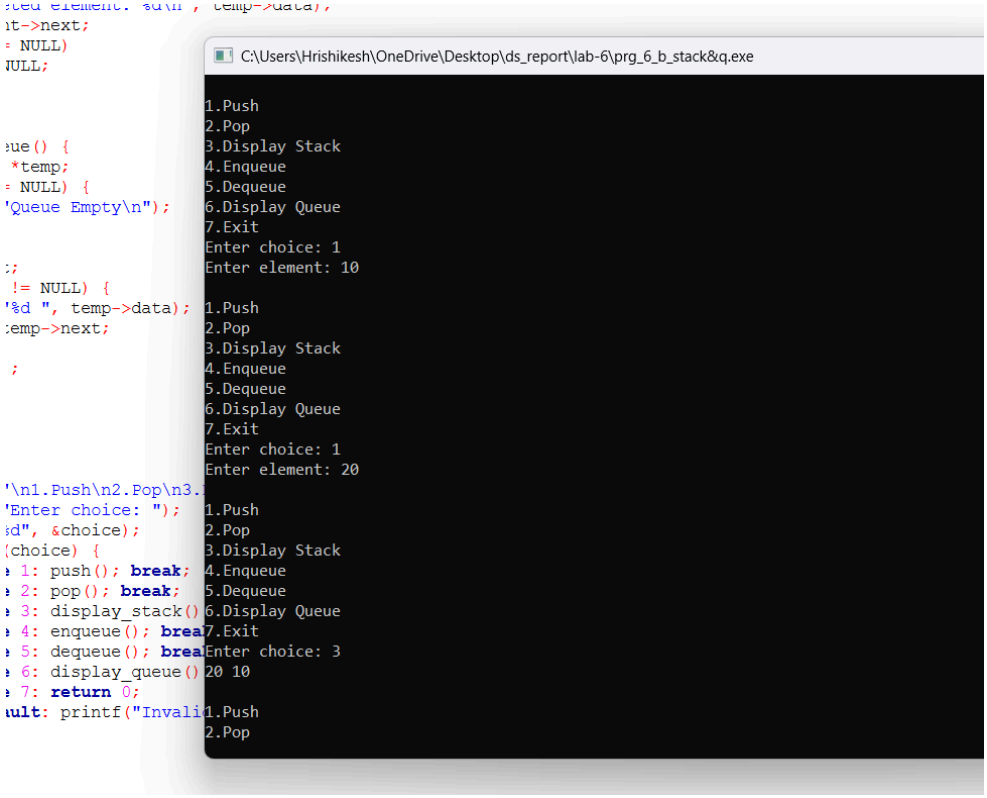
```
1.Create List 1
2.Create List 2
3.Sort List 1
4.Reverse List 1
5.Concatenate
6.Display List 1
7.Display List 2
8.Exit
Enter choice: 4

1.Create List 1
2.Create List 2
3.Sort List 1
4.Reverse List 1
5.Concatenate
6.Display List 1
7.Display List 2
8.Exit
Enter choice: 6
60 50 40 20 10

1.Create List 1
2.Create List 2
3.Sort List 1
4.Reverse List 1
5.Concatenate
6.Display List 1
7.Display List 2
8.Exit
Enter choice: 8

Process returned 0 (0x0)   execution time : 49.278 s
Press any key to continue.
```

LAB-6b (Stack & Queue simulation-SLL)

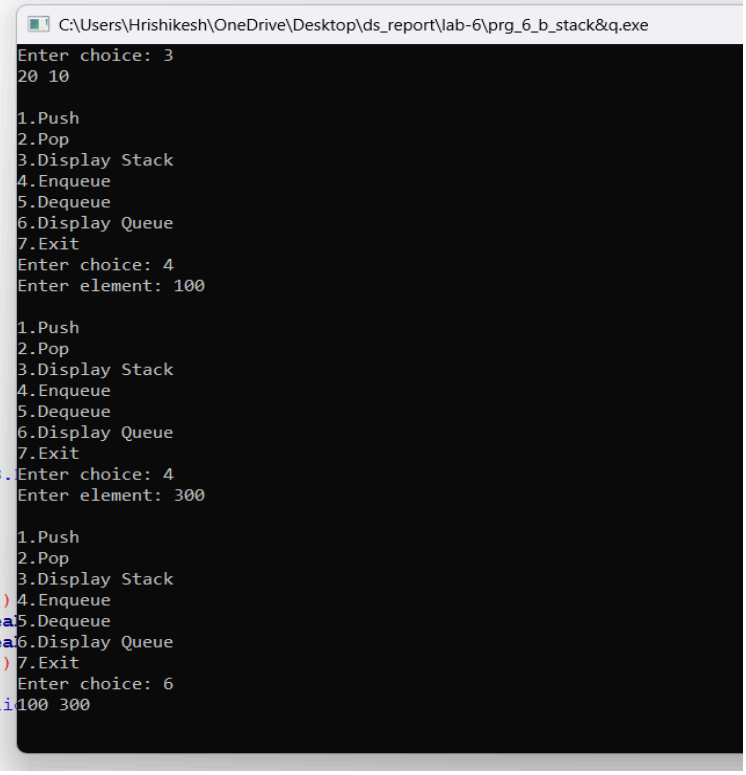
```c
#include <stdio.h>
#include <stdlib.h>

struct node {
    int data;
    struct node *next;
};

struct node *top = NULL;
struct node *front = NULL;
struct node *rear = NULL;

void push() {
    int val;
    struct node *newnode;
    newnode = (struct node *)malloc(sizeof(struct node));
    printf("Enter element: ");
    scanf("%d", &val);
    newnode->data = val;
    newnode->next = top;
    top = newnode;
}

void pop() {
    struct node *temp;
    if (top == NULL) {
        printf("Stack Empty\n");
        return;
    }
    temp = top;
    printf("Popped element: %d\n", temp->data);
    top = top->next;
    free(temp);
}

void display_stack() {
    struct node *temp;
    if (top == NULL) {
        printf("Stack Empty\n");
        return;
    }
    temp = top;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

void enqueue() {
    int val;
    struct node *newnode;
    newnode = (struct node *)malloc(sizeof(struct node));
    printf("Enter element: ");
    scanf("%d", &val);
    newnode->data = val;
    newnode->next = NULL;
    if (rear == NULL) {
        front = rear = newnode;
        return;
    }
    rear->next = newnode;
    rear = newnode;
}

void dequeue() {
    struct node *temp;
    if (front == NULL) {
        printf("Queue Empty\n");
        return;
    }
    temp = front;
    printf("Deleted element: %d\n", temp->data);
    front = front->next;
    if (front == NULL)
        rear = NULL;
    free(temp);
}
```

```c

void display_queue() {
    struct node *temp;
    if (front == NULL) {
        printf("Queue Empty\n");
        return;
    }
    temp = front;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

int main() {
    int choice;
    while (1) {
        printf("\n1.Push\n2.Pop\n3.Display Stack\n4.Enqueue\n5.Dequeue\n6.Display Queue\n7.Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1: push(); break;
            case 2: pop(); break;
            case 3: display_stack(); break;
            case 4: enqueue(); break;
            case 5: dequeue(); break;
            case 6: display_queue(); break;
            case 7: return 0;
            default: printf("Invalid choice\n");
        }
    }
}
```

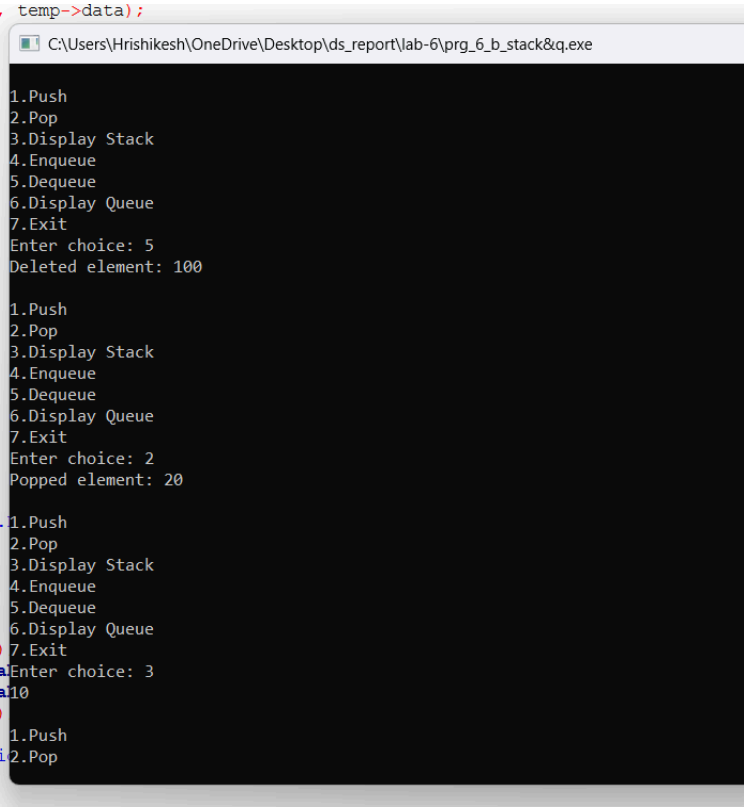
OUTPUT:



```
ext;
L)

 {
p;
L) {
e Empty\n");

JLL) {
, temp->data);
>next;

Push\n2.Pop\n3.
r choice: ");
&choice);
ce) {
push(); break;
pop(); break;
display_stack()
enqueue(); brea
dequeue(); brea
display_queue()
return 0;
 printf("Invali
```




```
element: %d\n", temp->data);
ext;
L)

 {
p;
L) {
e Empty\n");

JLL) {
, temp->data);
>next;

Push\n2.Pop\n3.
r choice: ");
&choice);
ce) {
push(); break;
pop(); break;
display_stack()
enqueue(); brea
dequeue(); brea
display_queue()
return 0;
 printf("Invali
```



```
xt;

[
:
 {
 Empty\n");


LL) {
 temp->data);
next;




ush\n2.Pop\n3.
 choice: ");
choice);
e) {
ush(); break;
op(); break;
isplay_stack()
nqueue(); brea
equeue(); brea
isplay_queue()
eturn 0;
printf("Invali
```

```
C:\Users\Hrishikesh\OneDrive\Desktop\ds_report\lab-6\prg_6_b_stack&q.exe
1.Push
2.Pop
3.Display Stack
4.Enqueue
5.Dequeue
6.Display Queue
7.Exit
Enter choice: 3
10

1.Push
2.Pop
3.Display Stack
4.Enqueue
5.Dequeue
6.Display Queue
7.Exit
Enter choice: 6
300

1.Push
2.Pop
3.Display Stack
4.Enqueue
5.Dequeue
6.Display Queue
7.Exit
Enter choice: 7

Process returned 0 (0x0)   execution time : 155.694 s
Press any key to continue.
```