

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

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  struct node {
5      int data;
6      struct node *next;
7  };
8
9  struct node *head1 = NULL;
10 struct node *head2 = NULL;
11
12 struct node* create(struct node *head) {
13     int n, i, val;
14     struct node *temp, *newnode;
15     printf("Enter number of nodes: ");
16     scanf("%d", &n);
17     for (i = 0; i < n; i++) {
18         newnode = (struct node *)malloc(sizeof(struct node));
19         printf("Enter data: ");
20         scanf("%d", &val);
21         newnode->data = val;
22         newnode->next = NULL;
23         if (head == NULL) {
24             head = newnode;
25             temp = head;
26         } else {
27             temp->next = newnode;
28             temp = newnode;
29         }
30     }
31     return head;
32 }
33
34 void display(struct node *head) {
35     struct node *temp;
36     if (head == NULL) {
37         printf("List is empty\n");
38         return;
39     }
40     temp = head;
41
42     while (temp != NULL) {
43         printf("%d ", temp->data);
44         temp = temp->next;
45     }
46     printf("\n");
47
48 void sort(struct node *head) {
49     struct node *i, *j;
50     int temp;
51     if (head == NULL)
52         return;
53     for (i = head; i->next != NULL; i = i->next) {
54         for (j = i->next; j != NULL; j = j->next) {
55             if (i->data > j->data) {
56                 temp = i->data;
57                 i->data = j->data;
58                 j->data = temp;
59             }
60         }
61     }
62 }
63
64 struct node* reverse(struct node *head) {
65     struct node *prev = NULL, *curr = head, *next;
66     while (curr != NULL) {
67         next = curr->next;
68         curr->next = prev;
69         prev = curr;
70         curr = next;
71     }
72     return prev;
73 }
74
75 struct node* concatenate(struct node *head1, struct node *head2) {
76     struct node *temp;
77     if (head1 == NULL)
78         return head2;
79     temp = head1;
80     while (temp->next != NULL)
```

## OUTPUT:

```
C:\Users\Ghishmiresh\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)

```

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 struct node *front = NULL;
11 struct node *rear = NULL;
12
13 void push() {
14     int val;
15     struct node *newnode;
16     newnode = (struct node *)malloc(sizeof(struct node));
17     printf("Enter element: ");
18     scanf("%d", &val);
19     newnode->data = val;
20     newnode->next = top;
21     top = newnode;
22 }
23
24 void pop() {
25     struct node *temp;
26     if (top == NULL) {
27         printf("Stack Empty\n");
28         return;
29     }
30     temp = top;
31     printf("Popped element: %d\n", temp->data);
32     top = top->next;
33     free(temp);
34 }
35
36 void display_stack() {
37     struct node *temp;
38     if (top == NULL) {
39         printf("Stack Empty\n");
40         return;
41     }
42     temp = top;
43     while (temp != NULL) {
44         printf("%d ", temp->data);
45         temp = temp->next;
46     }
47     printf("\n");
48 }
49
50 void enqueue() {
51     int val;
52     struct node *newnode;
53     newnode = (struct node *)malloc(sizeof(struct node));
54     printf("Enter element: ");
55     scanf("%d", &val);
56     newnode->data = val;
57     newnode->next = NULL;
58     if (rear == NULL) {
59         front = rear = newnode;
60         return;
61     }
62     rear->next = newnode;
63     rear = newnode;
64 }
65
66 void dequeue() {
67     struct node *temp;
68     if (front == NULL) {
69         printf("Queue Empty\n");
70         return;
71     }
72     temp = front;
73     printf("Deleted element: %d\n", temp->data);
74     front = front->next;
75     if (front == NULL)
76         rear = NULL;
77     free(temp);
78 }

```

```

79
80 void display_queue() {
81     struct node *temp;
82     if (front == NULL) {
83         printf("Queue Empty\n");
84         return;
85     }
86     temp = front;
87     while (temp != NULL) {
88         printf("%d ", temp->data);
89         temp = temp->next;
90     }
91     printf("\n");
92 }
93
94 int main() {
95     int choice;
96     while (1) {
97         printf("\n1.Push\n2.Pop\n3.Display Stack\n4.Enqueue\n5.Dequeue\n6.Display Queue\n7.Exit\n");
98         printf("Enter choice: ");
99         scanf("%d", &choice);
100         switch (choice) {
101             case 1: push(); break;
102             case 2: pop(); break;
103             case 3: display_stack(); break;
104             case 4: enqueue(); break;
105             case 5: dequeue(); break;
106             case 6: display_queue(); break;
107             case 7: return 0;
108             default: printf("Invalid choice\n");
109         }
110     }
111 }

```

## OUTPUT:

```

new element. sum\n , temp->data);
it->next;
= NULL)
NULL;

ue() {
*temp;
= NULL) {
'Queue Empty\n");

;;
!= NULL) {
'%d ', temp->data);
temp->next;

;

'\n1.Push\n2.Pop\n3.
'Enter choice: ");
%d", &choice);
(choice) {
1: push(); break;
2: pop(); break;
3: display_stack()
4: enqueue(); break;
5: dequeue(); break;
6: display_queue()
7: return 0;
ult: printf("Invalid

```

C:\Users\Hrshikesh\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: 1
Enter element: 10

1.Push
2.Pop
3.Display Stack
4.Enqueue
5.Dequeue
6.Display Queue
7.Exit
Enter choice: 1
Enter element: 20

1.Push
2.Pop
3.Display Stack
4.Enqueue
5.Dequeue
6.Display Queue
7.Exit
Enter choice: 3
Enter element: 20 10

1.Push
2.Pop

```

```

next;
L)

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

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

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

```

C:\Users\Hrshikesh\OneDrive\Desktop\ds\_report\lab-6\prg\_6\_b\_stack&q.exe

```

Enter choice: 3
20 10

1.Push
2.Pop
3.Display Stack
4.Enqueue
5.Dequeue
6.Display Queue
7.Exit
Enter choice: 4
Enter element: 100

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

1.Push
2.Pop
3.Display Stack
4.Enqueue
5.Dequeue
6.Display Queue
7.Exit
Enter choice: 6
100 300

```

```

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

```

C:\Users\Hrshikesh\OneDrive\Desktop\ds\_report\lab-6\prg\_6\_b\_stack&q.exe

```

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

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

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

```

```

1.Push
2.Pop
3.Display Stack
4.Enqueue
5.Dequeue
6.Display Queue
7.Exit
Enter choice: 5
Deleted element: 100

1.Push
2.Pop
3.Display Stack
4.Enqueue
5.Dequeue
6.Display Queue
7.Exit
Enter choice: 2
Popped element: 20

1.Push
2.Pop
3.Display Stack
4.Enqueue
5.Dequeue
6.Display Queue
7.Exit
Enter choice: 3
10

1.Push
2.Pop

```

```
ct;  
[
```

```
{  
:  
{  
Empty\n");
```

```
IL) {  
temp->data);  
next;
```

```
ish\n2.Pop\n3.
```

```
choice: ");  
choice);
```

```
3) {
```

```
ish()); break;
```

```
op(); break;
```

```
isplay_stack()
```

```
queue(); break;
```

```
queue(); break;
```

```
isplay_queue()
```

```
return 0;
```

```
printf("Invalid
```

```
C:\Users\Hrshikesh\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.
```