

```
Start here x lab6.c x linkedlistinsertion.c x lab4b.c x lab6final.c x
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  struct Node {
5      int data;
6      struct Node *next;
7  };
8
9  // Global heads
10 struct Node *head1 = NULL;
11 struct Node *head2 = NULL;
12
13 // Function prototypes
14 struct Node* createList(int n);
15 void displayList(struct Node *head);
16 void sortLinkedList(struct Node *head);
17 struct Node* reverseLinkedList(struct Node *head);
18 struct Node* concatenateLinkedList(struct Node *head1, struct Node *head2);
19
20 // Create list
21 struct Node* createList(int n) {
22     struct Node *head = NULL, *newNode, *temp;
23     int data;
24
25     if (n <= 0) {
26         printf("Number of nodes should be greater than 0\n");
27         return NULL;
28     }
29
30     for (int i = 1; i <= n; i++) {
31         newNode = (struct Node*)malloc(sizeof(struct Node));
32         if (newNode == NULL) {
33             printf("Memory allocation failed\n");
34             return head;
35         }
```

here X lab6.c X linkedlistinsertion.c X lab4b.c X lab6final.c X

```
34         return head;
35     }
36
37     printf("Enter data for node %d: ", i);
38     scanf("%d", &data);
39
40     newNode->data = data;
41     newNode->next = NULL;
42
43     if (head == NULL)
44         head = newNode;
45     else
46         temp->next = newNode;
47
48     temp = newNode;
49 }
50
51 printf("Linked list created successfully\n");
52 return head;
53 }
54
55 // Display
56 void displayList(struct Node *head) {
57     struct Node *temp = head;
58
59     if (head == NULL) {
60         printf("List is empty\n");
61         return;
62     }
63
64     printf("Linked List: ");
65     while (temp != NULL) {
66         printf("%d -> ", temp->data);
67         temp = temp->next;
68     }
```

```

67         temp = temp->next;
68     }
69     printf("NULL\n");
70 }
71
72 // Sort
73 void sortLinkedList(struct Node *head) {
74     struct Node *i, *j;
75     int tempData;
76
77     if (head == NULL) {
78         printf("List is empty, cannot sort.\n");
79         return;
80     }
81
82     for (i = head; i->next != NULL; i = i->next) {
83         for (j = i->next; j != NULL; j = j->next) {
84             if (i->data > j->data) {
85                 tempData = i->data;
86                 i->data = j->data;
87                 j->data = tempData;
88             }
89         }
90     }
91
92     printf("Linked list sorted successfully\n");
93 }
94
95 // Reverse
96 struct Node* reverseLinkedList(struct Node *head) {
97     struct Node *prev = NULL, *curr = head, *next = NULL;
98
99     while (curr != NULL) {
100         next = curr->next;
101         curr->next = prev;
102         prev = curr;
103         curr = next;
104     }
105
106     printf("Linked list reversed successfully\n");
107     return prev;
108 }
109
110 // Concatenate
111 struct Node* concatenateLinkedList(struct Node *head1, struct Node *head2) {
112     struct Node *temp;
113
114     if (head1 == NULL)
115         return head2;
116
117     temp = head1;
118     while (temp->next != NULL)
119         temp = temp->next;
120
121     temp->next = head2;
122
123     printf("Linked lists concatenated successfully\n");
124     return head1;
125 }
126
127 // Menu
128 int main() {
129     int choice, n;
130
131     while (1) {
132         printf("\n=====\\n");
133         printf("        LINKED LIST MENU        \\n");
134         printf("=====\\n");

```

```

100         next = curr->next;
101         curr->next = prev;
102         prev = curr;
103         curr = next;
104     }
105
106     printf("Linked list reversed successfully\n");
107     return prev;
108 }
109
110 // Concatenate
111 struct Node* concatenateLinkedList(struct Node *head1, struct Node *head2) {
112     struct Node *temp;
113
114     if (head1 == NULL)
115         return head2;
116
117     temp = head1;
118     while (temp->next != NULL)
119         temp = temp->next;
120
121     temp->next = head2;
122
123     printf("Linked lists concatenated successfully\n");
124     return head1;
125 }
126
127 // Menu
128 int main() {
129     int choice, n;
130
131     while (1) {
132         printf("\n=====\\n");
133         printf("        LINKED LIST MENU        \\n");
134         printf("=====\\n");

```

```
lab6.c X linkedlistinsertion.c X lab4b.c X lab6final.c X
printf("          LINKED LIST MENU          \n");
printf("===== \n");
printf("1. Create List 1\n");
printf("2. Create List 2\n");
printf("3. Display List 1\n");
printf("4. Display List 2\n");
printf("5. Sort List 1\n");
printf("6. Reverse List 1\n");
printf("7. Concatenate List 1 + List 2\n");
printf("8. Exit\n");
printf("Enter choice: ");
scanf("%d", &choice);

switch (choice) {
case 1:
    printf("Enter number of nodes for List 1: ");
    scanf("%d", &n);
    head1 = createList(n);
    break;

case 2:
    printf("Enter number of nodes for List 2: ");
    scanf("%d", &n);
    head2 = createList(n);
    break;

case 3:
    displayList(head1);
    break;

case 4:
    displayList(head2);
    break;

case 5:
```

tart here X lab6.c X linkedlistinsertion.c X lab4b.c X lab6final.c X

```
159         case 3:
160             displayList(head1);
161             break;
162
163         case 4:
164             displayList(head2);
165             break;
166
167         case 5:
168             sortLinkedList(head1);
169             break;
170
171         case 6:
172             head1 = reverseLinkedList(head1);
173             break;
174
175         case 7:
176             head1 = concatenateLinkedList(head1, head2);
177             printf("After concatenation:\n");
178             displayList(head1);
179             break;
180
181         case 8:
182             printf("Exiting program...\n");
183             exit(0);
184
185         default:
186             printf("Invalid choice. Try again.\n");
187     }
188 }
189
190 return 0;
191 }
192
193
```

```
=====
LINKED LIST MENU
=====
```

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

Enter choice: 1

Enter number of nodes for List 1: 3

Enter data for node 1: 5

Enter data for node 2: 2

Enter data for node 3: 9

Linked list created successfully

```
=====
LINKED LIST MENU
=====
```

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

Enter choice: 3

Linked List: 5 -> 2 -> 9 -> NULL

```
=====
LINKED LIST MENU
=====
```

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

Enter choice: 5

Linked list sorted successfully

```
=====
LINKED LIST MENU
=====
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Reverse List 1
7. Concatenate List 1 + List 2
8. Exit
Enter choice: 3
Linked List: 2 -> 5 -> 9 -> NULL
```

```
=====
LINKED LIST MENU
=====
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Reverse List 1
7. Concatenate List 1 + List 2
8. Exit
Enter choice: 6
Linked list reversed successfully
```

```
=====
LINKED LIST MENU
=====
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Reverse List 1
7. Concatenate List 1 + List 2
8. Exit
Enter choice: 3
Linked List: 9 -> 5 -> 2 -> NULL
```

```
=====
LINKED LIST MENU
=====
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
```

```

=====
                LINKED LIST MENU
=====
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Reverse List 1
7. Concatenate List 1 + List 2
8. Exit
Enter choice: 2
Enter number of nodes for List 2: 2
Enter data for node 1: 7
Enter data for node 2: 3
Linked list created successfully

=====
                LINKED LIST MENU
=====
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Reverse List 1
7. Concatenate List 1 + List 2
8. Exit
Enter choice: 7
Linked lists concatenated successfully
After concatenation:
Linked List: 9 -> 5 -> 2 -> 7 -> 3 -> NULL

=====
                LINKED LIST MENU
=====
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Reverse List 1
7. Concatenate List 1 + List 2
8. Exit
Enter choice: 8
Exiting program...

Process returned 0 (0x0)    execution time : 82.979 s
Press any key to continue.

```