

VARUN BHARGAVA – 241010282

DATA STRUCTURES TASK-4

Task 01: Reverse a Linked List:

(<https://github.com/varunnnb/dsa-sem3-iiitnr/blob/main/lab4/lab4-1.c>)

Write a program to:

1. Take input from the user to create a singly linked list.

```
C lab4-1.c U X
C lab4-1.c > main()
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  struct Node
5  {
6      int data;
7      struct Node *next;
8  };
9
10 int main()
11 {
12     struct Node *head = NULL, *tail = NULL;
13     int value;
14
15     printf("Enter values for the linked list (-1 to stop):\n");
16     while (1)
17     {
18         scanf("%d", &value);
19         if (value == -1)
20             break;
21
22         struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
23         newNode->data = value;
24         newNode->next = NULL;
25
26         if (head == NULL)
27         {
28             head = newNode;
29             tail = newNode;
30         }
31         else
32         {
33             tail->next = newNode;
34             tail = newNode;
35         }
36     }
37 }
```

2. Reverse the linked list.
3. Display the original and reversed linked list.

lab4-1.c U X

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct Node
5 {
6     int data;
7     struct Node *next;
8 };
9
10 struct Node *reverseList(struct Node *head)
11 {
12     struct Node *prev = NULL, *curr = head, *next = NULL;
13     while (curr != NULL)
14     {
15         next = curr->next;
16         curr->next = prev;
17         prev = curr;
18         curr = next;
19     }
20     return prev;
21 };
22
23 int main()
24 {
25     struct Node *head = NULL, *tail = NULL;
26     int value;
27
28     printf("Enter values for the linked list (-1 to stop):\n");
29     while (1)
30     {
31         scanf("%d", &value);
32         if (value == -1)
33             break;
34
35         struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
36         newNode->data = value;
37         newNode->next = NULL;
38
39         if (head == NULL)
40         {
41             head = newNode;
42             tail = newNode;
43         }
44         else
45         {
46             tail->next = newNode;
47             tail = newNode;
48         }
49     }
50
51     printf("linked list: ");
52     struct Node *temp = head;
53     while (temp != NULL)
54     {
55         printf("%d -> ", temp->data);
56         temp = temp->next;
57     }
58     printf("NULL\n");
59
60     head = reverseList(head);
61
62     printf("reversed linked list: ");
63     temp = head;
64     while (temp != NULL)
65     {
66         printf("%d -> ", temp->data);
67         temp = temp->next;
68     }
69     printf("NULL\n");
70
71     return 0;
72 }
73
```

CHAT

TERMINAL

```
c:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab4> gcc lab4-1.c -o lab4-1 ; if ($?) { gcc lab4-1.c -o lab4-1 ; if ($?) { .\lab4-1 }
Enter values for the linked list (-1 to stop):
32
12
423
linked list: 32 -> 12 -> 423 -> 456 -> 23 -> NULL
reversed linked list: 23 -> 456 -> 423 -> 12 -> 32 -> NULL
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab4>
```

lab4-1.c U X

```
23 int main()
24 {
25     while (1)
26     {
27         newnode->data = value;
28         newnode->next = NULL;
29
30         if (head == NULL)
31         {
32             head = newnode;
33             tail = newnode;
34         }
35         else
36         {
37             tail->next = newnode;
38             tail = newnode;
39         }
40     }
41
42     printf("linked list: ");
43     struct Node *temp = head;
44     while (temp != NULL)
45     {
46         printf("%d -> ", temp->data);
47         temp = temp->next;
48     }
49     printf("NULL\n");
50
51     head = reverseList(head);
52
53     printf("reversed linked list: ");
54     temp = head;
55     while (temp != NULL)
56     {
57         printf("%d -> ", temp->data);
58         temp = temp->next;
59     }
60     printf("NULL\n");
61
62     return 0;
63 }
64
```

CHAT

TERMINAL

```
c:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab4> gcc lab4-1.c -o lab4-1 ; if ($?) { gcc lab4-1.c -o lab4-1 ; if ($?) { .\lab4-1 }
Enter values for the linked list (-1 to stop):
32
12
423
linked list: 32 -> 12 -> 423 -> 456 -> 23 -> NULL
reversed linked list: 23 -> 456 -> 423 -> 12 -> 32 -> NULL
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab4>
```

main* Run Testcases 0 0 0 Ln 30, Col 6 Spaces: 4 UTF-8 CRLF C Go Live Win32 Prettier

Task 02: Check if Linked List is a Palindrome:

(<https://github.com/varunnnb/dsa-sem3-iiitnr/blob/main/lab4/lab4-2.c>)

Write a program to:

1. Take user input to create a singly linked list.

```
lab4-1.c U  lab4-2.c U X
lab4-2.c > ...
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  struct Node
5  {
6      int data;
7      struct Node *next;
8  };
9
10 int main()
11 {
12     struct Node *head = NULL, *tail = NULL;
13     int value;
14
15     printf("Enter values for the linked list (-1 to stop):\n");
16     while (1)
17     {
18         scanf("%d", &value);
19         if (value == -1)
20             break;
21
22         struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
23         newNode->data = value;
24         newNode->next = NULL;
25
26         if (head == NULL)
27         {
28             head = newNode;
29             tail = newNode;
30         }
31         else
32         {
33             tail->next = newNode;
34             tail = newNode;
35         }
36     }
37 }
```

2. Determine if the linked list is a palindrome (reads the same forward and backward).
3. Display an appropriate message indicating whether the linked list is a palindrome or not.

```
lab4-1.c U  C lab4-2.c U X
C lab4-2.c > main()
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct Node
5 {
6     int data;
7     struct Node *next;
8 };
9
10 int isPalindrome(struct Node *head)
11 {
12     int count = 0;
13     struct Node *temp = head;
14     while (temp != NULL)
15     {
16         count++;
17         temp = temp->next;
18     }
19
20     int *arr = (int *)malloc(count * sizeof(int));
21     temp = head;
22     for (int i = 0; i < count; i++)
23     {
24         arr[i] = temp->data;
25         temp = temp->next;
26     }
27
28     int flag = 1;
29     for (int i = 0, j = count - 1; i < j; i++, j--)
30     {
31         if (arr[i] != arr[j])
32         {
33             flag = 0;
34             break;
35         }
36     }
37     free(arr);
38     return flag;
39 }
40
41 int main()
42 {
43     struct Node *head = NULL, *tail = NULL;
44     int value;
45
46     printf("Enter values for the linked list (-1 to stop):\n");
47     while (1)
48     {
49         scanf("%d", &value);
50         if (value == -1)
51             break;
52
53         struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
54         newNode->data = value;
55         newNode->next = NULL;
56
57         if (head == NULL)
58         {
59             head = newNode;
60             tail = newNode;
61         }
62         else
63         {
64             tail->next = newNode;
65             tail = newNode;
66         }
67     }
68
69     printf("linked list: ");
70     struct Node *temp = head;
71     while (temp != NULL)
72     {
73         printf("%d -> ", temp->data);
74         temp = temp->next;
75     }
76
77     printf("NULL\n");
78
79     if (isPalindrome(head))
80         printf("The linked list is a palindrome.\n");
81     else
82         printf("The linked list is not a palindrome.\n");
83 }
```

```
lab4-1.c U  C lab4-2.c U X
C lab4-2.c > main()
41 int main()
42 {
43     struct Node *head = NULL, *tail = NULL;
44     int value;
45
46     printf("Enter values for the linked list (-1 to stop):\n");
47     while (1)
48     {
49         scanf("%d", &value);
50         if (value == -1)
51             break;
52
53         struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
54         newNode->data = value;
55         newNode->next = NULL;
56
57         if (head == NULL)
58         {
59             head = newNode;
60             tail = newNode;
61         }
62         else
63         {
64             tail->next = newNode;
65             tail = newNode;
66         }
67     }
68
69     printf("linked list: ");
70     struct Node *temp = head;
71     while (temp != NULL)
72     {
73         printf("%d -> ", temp->data);
74         temp = temp->next;
75     }
76
77     printf("NULL\n");
78
79     if (isPalindrome(head))
80         printf("The linked list is a palindrome.\n");
81     else
82         printf("The linked list is not a palindrome.\n");
83 }
```

```
CHAT
TERMINAL
linked list: 32 -> 12 -> 423 -> 456 -> 23 -> NULL
reversed linked list: 23 -> 456 -> 423 -> 12 -> 32 -> NULL
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab4> cd c:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab4; if ($?) { gcc lab4-2.c -o lab4-2 } ; if ($?) { .\lab4-2 }
Enter values for the linked list (-1 to stop):
32
41
421
456
45
654
-1
linked list: 32 -> 41 -> 421 -> 456 -> 45 -> 654 -> NULL
The linked list is not a palindrome.
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab4>

CHAT
TERMINAL
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab4> cd c:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab4; if ($?) { gcc lab4-2.c -o lab4-2 } ; if ($?) { .\lab4-2 }
Enter values for the linked list (-1 to stop):
32
23
14
23
32
-1
linked list: 32 -> 23 -> 14 -> 23 -> 32 -> NULL
The linked list is a palindrome.
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab4>
```

Task 03: Merge Two Linked Lists:

(<https://github.com/varunnrb/dsa-sem3-iiitr/blob/main/lab4/lab4-3.c>)

Write a program to:

1. Take user input to create two unsorted singly linked lists.
2. Merge the two linked lists in ascending order into a single sorted linked list.
3. Display the final merged sorted linked list.

```
lab4-3.c > mergeSortedLists(Node*, Node*)
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct Node
5 {
6     int data;
7     struct Node *next;
8 };
9
10 struct Node *createList()
11 {
12     struct Node *head = NULL, *tail = NULL;
13     int value;
14     printf("Enter values for the linked list (-1 to stop):\n");
15     while (1)
16     {
17         scanf("%d", &value);
18         if (value == -1)
19             break;
20         struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
21         newNode->data = value;
22         newNode->next = NULL;
23         if (head == NULL)
24         {
25             head = newNode;
26             tail = newNode;
27         }
28         else
29         {
30             tail->next = newNode;
31             tail = newNode;
32         }
33     }
34     return head;
35 }
36
37 void printList(struct Node *head)
38 {
39     struct Node *temp = head;
40     while (temp != NULL)
41     {
42         printf("%d -> ", temp->data);
43         temp = temp->next;
44     }
45     printf("NULL\n");
46 }
47
48 void sortList(struct Node *head)
49 {
50     if (!head)
51         return;
52     int swapped;
53     struct Node *ptr1, *lptr = NULL;
54     do
55     {
56         swapped = 0;
57         ptr1 = head;
58         while (ptr1->next != lptr)
59         {
60             if (ptr1->data > ptr1->next->data)
61             {
62                 int temp = ptr1->data;
63                 ptr1->data = ptr1->next->data;
64                 ptr1->next->data = temp;
65                 swapped = 1;
66             }
67             ptr1 = ptr1->next;
68         }
69         lptr = ptr1;
70     } while (swapped);
71 }
72
73 struct Node *mergeSortedLists(struct Node *l1, struct Node *l2)
74 {
75     struct Node dummy;
76     struct Node *tail = &dummy;
77     dummy->next = NULL;
78
79     while (l1 != NULL || l2 != NULL)
80     {
81         if (l1 == NULL)
82             tail->next = l2;
83         else if (l2 == NULL)
84             tail->next = l1;
85         else if (l1->data < l2->data)
86             tail->next = l1;
87         else
88             tail->next = l2;
89         if (l1 != NULL)
90             l1 = l1->next;
91         if (l2 != NULL)
92             l2 = l2->next;
93         tail = tail->next;
94     }
95     tail->next = NULL;
96     return dummy->next;
97 }
```

```
cd "c:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab4\" ; if ($?) { gcc lab4-3.c -o lab4-3 } ; if ($?) { .\lab4-3 }
Create first linked list:
Enter values for the linked list (-1 to stop):
32
1234
423
5
6
0
-1
Create second linked list:
Enter values for the linked list (-1 to stop):
31
345
1
56
89
-1
First sorted linked list: 0 -> 5 -> 6 -> 32 -> 423 -> 1234 -> NULL
Second sorted linked list: 1 -> 31 -> 56 -> 89 -> 345 -> NULL
Merged sorted linked list: 0 -> 1 -> 5 -> 6 -> 31 -> 32 -> 56 -> 89 -> 345 -> 423 -> 1234 -> NULL
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab4>
```

```
lab4-3.c > mergeSortedLists(Node*, Node*)
37 void printList(struct Node *head)
38 {
39     struct Node *temp = head;
40     while (temp != NULL)
41     {
42         printf("%d -> ", temp->data);
43         temp = temp->next;
44     }
45     printf("NULL\n");
46 }
47
48 void sortList(struct Node *head)
49 {
50     if (!head)
51         return;
52     int swapped;
53     struct Node *ptr1, *lptr = NULL;
54     do
55     {
56         swapped = 0;
57         ptr1 = head;
58         while (ptr1->next != lptr)
59         {
60             if (ptr1->data > ptr1->next->data)
61             {
62                 int temp = ptr1->data;
63                 ptr1->data = ptr1->next->data;
64                 ptr1->next->data = temp;
65                 swapped = 1;
66             }
67             ptr1 = ptr1->next;
68         }
69         lptr = ptr1;
70     } while (swapped);
71 }
72
73 struct Node *mergeSortedLists(struct Node *l1, struct Node *l2)
74 {
75     struct Node dummy;
76     struct Node *tail = &dummy;
77     dummy->next = NULL;
78
79     while (l1 != NULL || l2 != NULL)
80     {
81         if (l1 == NULL)
82             tail->next = l2;
83         else if (l2 == NULL)
84             tail->next = l1;
85         else if (l1->data < l2->data)
86             tail->next = l1;
87         else
88             tail->next = l2;
89         if (l1 != NULL)
90             l1 = l1->next;
91         if (l2 != NULL)
92             l2 = l2->next;
93         tail = tail->next;
94     }
95     tail->next = NULL;
96     return dummy->next;
97 }
```

```
cd "c:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab4\" ; if ($?) { gcc lab4-3.c -o lab4-3 } ; if ($?) { .\lab4-3 }
Create first linked list:
Enter values for the linked list (-1 to stop):
32
1234
423
5
6
0
-1
Create second linked list:
Enter values for the linked list (-1 to stop):
31
345
1
56
89
-1
First sorted linked list: 0 -> 5 -> 6 -> 32 -> 423 -> 1234 -> NULL
Second sorted linked list: 1 -> 31 -> 56 -> 89 -> 345 -> NULL
Merged sorted linked list: 0 -> 1 -> 5 -> 6 -> 31 -> 32 -> 56 -> 89 -> 345 -> 423 -> 1234 -> NULL
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab4>
```

```
lab4-1.c U lab4-2.exe U lab4-2.c U lab4-3.c U X
C lab4-3.c > mergeSortedLists(Node*, Node*)
73 struct Node *mergeSortedLists(struct Node *l1, struct Node *l2)
74 {
75     struct Node dummy;
76     struct Node *tail = &dummy;
77     dummy.next = NULL;
78
79     while (l1 && l2)
80     {
81         if (l1->data < l2->data)
82         {
83             tail->next = l1;
84             l1 = l1->next;
85         }
86         else
87         {
88             tail->next = l2;
89             l2 = l2->next;
90         }
91         tail = tail->next;
92     }
93     if (l1)
94         tail->next = l1;
95     if (l2)
96         tail->next = l2;
97     return dummy.next;
98 }
99
100 int main()
101 {
102     printf("Create first linked list:\n");
103     struct Node *head1 = createlist();
104
105     printf("Create second linked list:\n");
106     struct Node *head2 = createlist();
107
108     sortList(head1);
109     sortList(head2);
110
111     printf("First sorted linked list: ");
112     printList(head1);
113
114     printf("Second sorted linked list: ");
115     printList(head2);
116
117     struct Node *mergedHead = mergeSortedLists(head1, head2);
118
119     printf("Merged sorted linked list: ");
120     printList(mergedHead);
121
122     return 0;
123 }
```

```
cd "c:\Users\varun\Desktop\VB\College\IIITNR\assigments\sem3\dsa\lab4"
; if ($?) { gcc lab4-3.c -o lab4-3 } ; if ($?) { .\lab4-3 }
Create first linked list:
Enter values for the linked list (-1 to stop):
32
1234
423
5
6
0
-1
Create second linked list:
Enter values for the linked list (-1 to stop):
31
345
1
56
89
-1
First sorted linked list: 0 -> 5 -> 6 -> 32 -> 423 -> 1234 -> NULL
Second sorted linked list: 1 -> 31 -> 56 -> 89 -> 345 -> NULL
Merged sorted linked list: 0 -> 1 -> 5 -> 6 -> 31 -> 32 -> 56 -> 89 ->
345 -> 423 -> 1234 -> NULL
PS C:\Users\varun\Desktop\VB\College\IIITNR\assigments\sem3\dsa\lab4>
```

```
lab4-1.c U lab4-2.exe U lab4-2.c U lab4-3.c U X
C lab4-3.c > mergeSortedLists(Node*, Node*)
73 struct Node *mergeSortedLists(struct Node *l1, struct Node *l2)
79 while (l1 && l2)
93 if (l1)
94     tail->next = l1;
95 if (l2)
96     tail->next = l2;
97 return dummy.next;
98 }
99
100 int main()
101 {
102     printf("Create first linked list:\n");
103     struct Node *head1 = createlist();
104
105     void sortList(struct Node *head) "\n";
106     Generate Copilot summary
107     sortList(head1);
108     sortList(head2);
109
110     printf("First sorted linked list: ");
111     printList(head1);
112
113     printf("Second sorted linked list: ");
114     printList(head2);
115
116     struct Node *mergedHead = mergeSortedLists(head1, head2);
117
118     printf("Merged sorted linked list: ");
119     printList(mergedHead);
120
121     return 0;
122 }
123 }
```

```
cd "c:\Users\varun\Desktop\VB\College\IIITNR\assigments\sem3\dsa\lab4"
; if ($?) { gcc lab4-3.c -o lab4-3 } ; if ($?) { .\lab4-3 }
Create first linked list:
Enter values for the linked list (-1 to stop):
32
1234
423
5
6
0
-1
Create second linked list:
Enter values for the linked list (-1 to stop):
31
345
1
56
89
-1
First sorted linked list: 0 -> 5 -> 6 -> 32 -> 423 -> 1234 -> NULL
Second sorted linked list: 1 -> 31 -> 56 -> 89 -> 345 -> NULL
Merged sorted linked list: 0 -> 1 -> 5 -> 6 -> 31 -> 32 -> 56 -> 89 ->
345 -> 423 -> 1234 -> NULL
PS C:\Users\varun\Desktop\VB\College\IIITNR\assigments\sem3\dsa\lab4>
```