

LAB PROGRAM - 6

a). SLL - Delete (start,end,any position)

```
Start here X LinkedlistDeletion.c X LinkedlistSorting.c 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 *head = NULL;
11 struct Node* createNode(int data)
12 {
13     struct Node *newNode = (struct Node*) malloc(sizeof(struct Node));
14     newNode->data = data;
15     newNode->next = NULL;
16     return newNode;
17 }
18
19 void insertEnd(int data)
20 {
21     struct Node *newNode = createNode(data);
22     if (head == NULL)
23     {
24         head = newNode;
25         return;
26     }
27     struct Node *temp = head;
28     while (temp->next != NULL)
29         temp = temp->next;
30     temp->next = newNode;
31 }
32
33 void deleteBeginning()
34 {
35     if (head == NULL)
36     {
37         printf("List is empty!\n");
38         return;
39     }
40     struct Node *temp = head;
41     head = head->next;
42     printf("Deleted element: %d\n", temp->data);
43     free(temp);
44 }
45
```

```
Start here X LinkedlistDeletion.c X LinkedlistSorting.c X
46 void deleteEnd()
47 {
48     if (head == NULL)
49     {
50         printf("List is empty!\n");
51         return;
52     }
53     struct Node *temp = head, *prev = NULL;
54     if (head->next == NULL)
55     {
56         printf("Deleted element: %d\n", head->data);
57         free(head);
58         head = NULL;
59         return;
60     }
61     while (temp->next != NULL)
62     {
63         prev = temp;
64         temp = temp->next;
65     }
66     printf("Deleted element: %d\n", temp->data);
67     prev->next = NULL;
68     free(temp);
69 }
70
71 void deleteAtPosition(int pos)
72 {
73     if (head == NULL)
74     {
75         printf("List is empty!\n");
76         return;
77     }
78     if (pos == 1)
79     {
80         deleteBeginning();
81         return;
82     }
83     struct Node *temp = head, *prev = NULL;
84     for (int i = 1; i < pos && temp != NULL; i++)
85     {
86         prev = temp;
87         temp = temp->next;
88     }
89 }
90
```

```
Start here X LinkedListDeletion.c X LinkedListSorting.c X
82
83
84 struct Node *temp = head, *prev = NULL;
85 for (int i = 1; i < pos && temp != NULL; i++)
86 {
87     prev = temp;
88     temp = temp->next;
89 }
90
91 if (temp == NULL)
92 {
93     printf("Position not found!\n");
94     return;
95 }
96
97 prev->next = temp->next;
98 printf("Deleted element: %d\n", temp->data);
99 free(temp);
100
101
102 void display()
103 {
104     if (head == NULL)
105     {
106         printf("List is empty!\n");
107         return;
108     }
109     struct Node *temp = head;
110     printf("Linked List: ");
111     while (temp != NULL)
112     {
113         printf("%d -> ", temp->data);
114         temp = temp->next;
115     }
116     printf("NULL\n");
117 }
118
119 int main()
120 {
121     int choice, data, pos;
122     while (1)
123     {
124         printf("\n--- Linked List Deletion Menu ---\n");
125         printf("1. Insert End\n");
126         printf("2. Delete Beginning\n");
```

```
Start here X LinkedListDeletion.c X LinkedListSorting.c X
119 int main()
120 {
121     int choice, data, pos;
122     while (1)
123     {
124         printf("\n--- Linked List Deletion Menu ---\n");
125         printf("1. Insert End\n");
126         printf("2. Delete Beginning\n");
127         printf("3. Delete End\n");
128         printf("4. Delete at Position\n");
129         printf("5. Display\n");
130         printf("6. Exit\n");
131         printf("Enter your choice: ");
132         scanf("%d", &choice);
133
134         switch (choice)
135         {
136             case 1:
137                 printf("Enter data: ");
138                 scanf("%d", &data);
139                 insertEnd(data);
140                 break;
141             case 2:
142                 deleteBeginning();
143                 break;
144             case 3:
145                 deleteEnd();
146                 break;
147             case 4:
148                 printf("Enter position to delete: ");
149                 scanf("%d", &pos);
150                 deleteAtPosition(pos);
151                 break;
152             case 5:
153                 display();
154                 break;
155             case 6:
156                 exit(0);
157             default:
158                 printf("Invalid choice!\n");
159         }
160     }
161     return 0;
162 }
163
```

OUTPUT:

```
D:\MF24C3067\LinkedListDel X + v
---- Linked List Deletion Menu ----
1. Insert End
2. Delete Beginning
3. Delete End
4. Delete at Position
5. Display
6. Exit
Enter your choice: 1
Enter data: 11

---- Linked List Deletion Menu ----
1. Insert End
2. Delete Beginning
3. Delete End
4. Delete at Position
5. Display
6. Exit
Enter your choice: 1
Enter data: 12

---- Linked List Deletion Menu ----
1. Insert End
2. Delete Beginning
3. Delete End
4. Delete at Position
5. Display
6. Exit
Enter your choice: 1
Enter data: 23

---- Linked List Deletion Menu ----
1. Insert End
2. Delete Beginning
3. Delete End
4. Delete at Position
5. Display
6. Exit
Enter your choice: 5
Linked List: 11 -> 12 -> 23 -> NULL

---- Linked List Deletion Menu ----
1. Insert End
2. Delete Beginning
3. Delete End
4. Delete at Position
5. Display
6. Exit
Enter your choice: 2
Deleted element: 11

Finance headline
India reported 8...
```

```
D:\MF24C3067\LinkedListDel X + v
1. Insert End
2. Delete Beginning
3. Delete End
4. Delete at Position
5. Display
6. Exit
Enter your choice: 5
Linked List: 11 -> 12 -> 23 -> NULL

---- Linked List Deletion Menu ----
1. Insert End
2. Delete Beginning
3. Delete End
4. Delete at Position
5. Display
6. Exit
Enter your choice: 2
Deleted element: 11

---- Linked List Deletion Menu ----
1. Insert End
2. Delete Beginning
3. Delete End
4. Delete at Position
5. Display
6. Exit
Enter your choice: 3
Deleted element: 23

---- Linked List Deletion Menu ----
1. Insert End
2. Delete Beginning
3. Delete End
4. Delete at Position
5. Display
6. Exit
Enter your choice: 5
Linked List: 12 -> NULL

---- Linked List Deletion Menu ----
1. Insert End
2. Delete Beginning
3. Delete End
4. Delete at Position
5. Display
6. Exit
Enter your choice: 6

Process returned 0 (0x0)   execution time : 135.353 s
Press any key to continue.
```

b). SLL - (sort, reverse, concatenation)

```
Start here X LinkedListDeletion.c X LinkedListSorting.c 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 *reverse(struct Node *start)
11 {
12     struct Node *prev = NULL;
13     struct Node *curr = start;
14     struct Node *next;
15     while (curr != NULL)
16     {
17         next = curr->next;
18         curr->next = prev;
19         prev = curr;
20         curr = next;
21     }
22     return prev;
23 }
24
25 struct Node *concatenate(struct Node *start1, struct Node *start2)
26 {
27     struct Node *ptr;
28     if (start1 == NULL)
29     {
30         return start2;
31     }
32     ptr = start1;
33     while (ptr->next != NULL)
34     {
35         ptr = ptr->next;
36     }
37     ptr->next = start2;
38     return start1;
39 }
40
41 struct Node *sort_list(struct Node *start)
42 {
43     struct Node *i, *j;
```

```
Start here X LinkedListDeletion.c X LinkedListSorting.c X
41 struct Node *sort_list(struct Node *start)
42 {
43     struct Node *i, *j;
44     int temp;
45     if (start == NULL)
46         return start;
47
48     for (i = start; i->next != NULL; i = i->next)
49     {
50         for (j = i->next; j != NULL; j = j->next)
51         {
52             if (i->data > j->data)
53             {
54                 temp = i->data;
55                 i->data = j->data;
56                 j->data = temp;
57             }
58         }
59     }
60     return start;
61 }
62
63 struct Node* createNode(int data)
64 {
65     struct Node *newNode = (struct Node*)malloc(sizeof(struct Node));
66     newNode->data = data;
67     newNode->next = NULL;
68     return newNode;
69 }
70
71 void insertEnd(struct Node **head, int data)
72 {
73     struct Node *newNode = createNode(data);
74     if (*head == NULL) {
75         *head = newNode;
76         return;
77     }
78     struct Node *temp = *head;
79     while (temp->next != NULL)
80         temp = temp->next;
81     temp->next = newNode;
82 }
83 }
```

```
Start here X LinkedListDeletion.c X LinkedListSorting.c X
84 void display(struct Node *head)
85 {
86     if (head == NULL)
87     {
88         printf("List is empty!\n");
89         return;
90     }
91     struct Node *temp = head;
92     while (temp != NULL)
93     {
94         printf("%d -> ", temp->data);
95         temp = temp->next;
96     }
97     printf("NULL\n");
98 }
99
100 int main()
101 {
102     struct Node *head1 = NULL, *head2 = NULL;
103
104     //List 1
105     insertEnd(&head1, 10);
106     insertEnd(&head1, 30);
107     insertEnd(&head1, 20);
108     insertEnd(&head1, 40);
109
110     //List 2
111     insertEnd(&head2, 15);
112     insertEnd(&head2, 5);
113     insertEnd(&head2, 25);
114     insertEnd(&head2, 35);
115
116     printf("\nInitial Lists:\n");
117     printf("List 1: ");
118     display(head1);
119     printf("List 2: ");
120     display(head2);
121
122     printf("\nAfter Sorting:\n");
123     head1 = sort_list(head1);
124     head2 = sort_list(head2);
125     printf("List 1 (sorted): ");
126     display(head1);
127     printf("List 2 (sorted): ");
128     display(head2);
129 }
```

C/C++ Windows (CR+LF) WINDOWS-1252 Line 5, Col 1, Pos 56 Insert Read/Write default 11:53:38 13-11-2023

```
Start here X LinkedListDeletion.c X LinkedListSorting.c X
99
100 int main()
101 {
102     struct Node *head1 = NULL, *head2 = NULL;
103
104     //List 1
105     insertEnd(&head1, 10);
106     insertEnd(&head1, 30);
107     insertEnd(&head1, 20);
108     insertEnd(&head1, 40);
109
110     //List 2
111     insertEnd(&head2, 15);
112     insertEnd(&head2, 5);
113     insertEnd(&head2, 25);
114     insertEnd(&head2, 35);
115
116     printf("\nInitial Lists:\n");
117     printf("List 1: ");
118     display(head1);
119     printf("List 2: ");
120     display(head2);
121
122     printf("\nAfter Sorting:\n");
123     head1 = sort_list(head1);
124     head2 = sort_list(head2);
125     printf("List 1 (sorted): ");
126     display(head1);
127     printf("List 2 (sorted): ");
128     display(head2);
129
130     printf("\nAfter Reversing:\n");
131     head1 = reverse(head1);
132     head2 = reverse(head2);
133     printf("List 1 (reversed): ");
134     display(head1);
135     printf("List 2 (reversed): ");
136     display(head2);
137
138     printf("\nAfter Concatenation:\n");
139     head1 = concatenate(head1, head2);
140     printf("Concatenated List: ");
141     display(head1);
142
143     return 0;
144 }
145
```

C/C++ Windows (CR+LF) WINDOWS-1252 Line 5, Col 1, Pos 56 Insert Read/Write default 11:54:28 13-11-2023

OUTPUT:

```
D:\1BF24CS067\LinkedListSort >
Initial Lists:
List 1: 10 -> 30 -> 20 -> 40 -> NULL
List 2: 15 -> 5 -> 25 -> 35 -> NULL

After Sorting:
List 1 (sorted): 10 -> 20 -> 30 -> 40 -> NULL
List 2 (sorted): 5 -> 15 -> 25 -> 35 -> NULL

After Reversing:
List 1 (reversed): 40 -> 30 -> 20 -> 10 -> NULL
List 2 (reversed): 35 -> 25 -> 15 -> 5 -> NULL

After Concatenation:
Concatenated List: 40 -> 30 -> 20 -> 10 -> 35 -> 25 -> 15 -> 5 -> NULL

Process returned 0 (0x0)   execution time : 0.005 s
Press any key to continue.
|
```