

Laboratory 7

1. Questions

1. Implement a linked list and illustrate the following operations.
 - i. Insert a node at the beginning
 - ii. Insert a node at the end
 - iii. Print the linked list
2. Write a program to create a linked list and delete the element entered by a user.

2. Program

1.1)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  struct node {
4      int data;
5      struct node *next;
6  } *head;
7
8  void createList(int n);
9  void insertNodeAtBeginning(int data);
10 void displayList();
11
12 int main()
13 {
14     int n, data;
15     printf("Enter the total number of nodes: \n");
16     scanf("%d", &n);
17     createList(n);
18
19     printf("\nData in the list \n");
20     displayList();
21     printf("\nEnter data to insert at beginning of the list: ");
22     scanf("%d", &data);
23     insertNodeAtBeginning(data);
24
25     printf("\nData in the list \n");
26     displayList();
27
28     return 0;
29 }
```

```
31 void createList(int n)
32 {
33     struct node *newNode, *temp;
34     int data, i;
35     head = (struct node *)malloc(sizeof(struct node));
36     if(head == NULL){
37         printf("Unable to allocate memory.");
38     }
39     else{
40         printf("Enter the data of node 1: ");
41         scanf("%d", &data);
42
43         head->data = data; // Link data field with data
44         head->next = NULL; // Link address field to NULL
45
46         temp = head;
47         for(i=2; i<=n; i++){
48             newNode = (struct node *)malloc(sizeof(struct node));
49             if(newNode == NULL){
50                 printf("Unable to allocate memory.");
51                 break;
52             }
53
54             else{
55                 printf("Enter the data of node %d: ", i);
56                 scanf("%d", &data);
57
58                 newNode->data = data; // Link data field of newNode with data
59                 newNode->next = NULL; // Link address field of newNode with NULL
60
61                 temp->next = newNode; // Link previous node i.e. temp to the newNode
62
63                 temp = temp->next;
64             }
65         }
66         printf("SINGLY LINKED LIST CREATED SUCCESSFULLY\n");
67     }
```

```
68 void insertNodeAtBeginning(int data)
69 {
70     struct node *newNode;
71
72     newNode = (struct node*)malloc(sizeof(struct node));
73
74     if(newNode == NULL)
75     {
76         printf("Unable to allocate memory.");
77     }
78     else
79     {
80         newNode->data = data; // Link data part
81         newNode->next = head; // Link address part
82
83         head = newNode; // Make newNode as first node
84
85         printf("DATA INSERTED SUCCESSFULLY\n");
86     }
87 }
88
```

```
89 void displayList()
90 {
91     struct node *temp;
92     if(head == NULL)
93     {
94         printf("List is empty.");
95     }
96     else
97     {
98         printf("data of list is: \n");
99         temp = head;
100         while(temp != NULL)
101         {
102             printf("%d ", temp->data); // Print data of current node
103             temp = temp->next; // Move to next node
104         }
105     }
106 }
```

1.2)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  struct node {
4      int data;          // Data
5      struct node *next; // Address
6  } *head;
7
8  void createList(int n);
9  void insertNodeAtEnd(int data);
10 void displayList();
11
12 int main()
13 {
14     int n, data;
15     printf("Enter the total number of nodes: ");
16     scanf("%d", &n);
17     createList(n);
18
19     printf("\nData in the list \n");
20     displayList();
21     printf("\nEnter data to insert at end of the list: ");
22     scanf("%d", &data);
23     insertNodeAtEnd(data);
```

```
24
25     printf("\nData in the list \n");
26     displayList();
27
28     return 0;
29 }
30
31 /*
32  * Create a list of n nodes
33  */
34 void createList(int n)
35 {
36     struct node *newNode, *temp;
37     int data, i;
38
39     head = (struct node *)malloc(sizeof(struct node));
40
41     /*
42      * If unable to allocate memory for head node
43      */
44     if(head == NULL)
45     {
46         printf("Unable to allocate memory.");
47     }
48     else
49     {
50         /*
51          * Reads data of node from the user
52          */
53         printf("Enter the data of node 1: ");
54         scanf("%d", &data);
55
56         head->data = data; // Link the data field with data
57         head->next = NULL; // Link the address field to NULL
58
59         temp = head;
60
61         /*
62          * Create n nodes and adds to linked list
63          */
64         for(i=2; i<=n; i++)
65         {
66             newNode = (struct node *)malloc(sizeof(struct node));
67
```

```
68      /* If memory is not allocated for newNode */
69      if(newNode == NULL)
70      {
71          printf("Unable to allocate memory.");
72          break;
73      }
74      else
75      {
76          printf("Enter the data of node %d: ", i);
77          scanf("%d", &data);
78
79          newNode->data = data; // Link the data field of newNode with data
80          newNode->next = NULL; // Link the address field of newNode with NULL
81
82          temp->next = newNode; // Link previous node i.e. temp to the newNode
83          temp = temp->next;
84      }
85  }
86
87  printf("SINGLY LINKED LIST CREATED SUCCESSFULLY\n");
88  }
89  }
90
```

Activate Windows

```
90
91 void insertNodeAtEnd(int data)
92 {
93     struct node *newNode, *temp;
94
95     newNode = (struct node*)malloc(sizeof(struct node));
96
97     if(newNode == NULL)
98     {
99         printf("Unable to allocate memory.");
100     }
101     else
102     {
103         newNode->data = data; // Link the data part
104         newNode->next = NULL;
105
106         temp = head;
107
108         // Traverse to the last node
109         while(temp->next != NULL)
110             temp = temp->next;
111
112         temp->next = newNode; // Link address part

```

Activate Windows

```
112         temp->next = newNode; // Link address part
113
114         printf("DATA INSERTED SUCCESSFULLY\n");
115     }
116 }
117
118 /*
119  * Display entire list
120  */
121 void displayList()
122 {
123     struct node *temp;
124
125     /*
126      * If the list is empty i.e. head = NULL
127      */
128     if(head == NULL)
129     {
130         printf("List is empty.");
131     }
132     else
133     {
134         printf("list is :-\n");
135         temp = head;
136         while(temp != NULL)
137         {
138             printf(" %d ", temp->data); // Print data of current node
139             temp = temp->next;           // Move to next node
140         }
141     }
142 }
```

1.3)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  struct Node
4  {
5      int data;
6      struct Node *next;
7  };
8
```

```
9  void deleteNode(struct Node *head, struct Node *n)
10 {
11     if(head == n) {
12         if(head->next == NULL){
13             printf("There is only one node. The list can't be made empty ");
14             return;
15         }
16         head->data = head->next->data;
17         n = head->next;
18         head->next = head->next->next;
19         // free memory
20         free(n);
21
22         return;
23     }
24     struct Node *prev = head;
25     while(prev->next != NULL && prev->next != n)
26         prev = prev->next;
27     if(prev->next == NULL){
28         printf("\n Given node is not present in Linked List");
29         return;
30     }
31     prev->next = prev->next->next;
32     free(n);
33     return;
34 }
```



```
36  /* Utility function to insert a node at the beginning */
37  void push(struct Node **head_ref, int new_data)
38  {
39      struct Node *new_node =
40          (struct Node *)malloc(sizeof(struct Node));
41      new_node->data = new_data;
42      new_node->next = *head_ref;
43      *head_ref = new_node;
44  }
45
46  /* Utility function to print a linked list */
47  void printList(struct Node *head)
48  {
49      while(head!=NULL)
50      {
51          printf("%d ",head->data);
52          head=head->next;
53      }
54      printf("\n");
55  }
56
```

```
56  int main()
57  {
58      struct Node *head = NULL;
59      push(&head, 10);
60      push(&head, 22);
61      push(&head, 34);
62      push(&head, 49);
63
64      printf("Given Linked List: ");
65      printList(head);
66      printf("\nDeleting node %d: ", head->next->next->data);
67      deleteNode(head, head->next->next);
68
69      printf("\nModified Linked List: ");
70      printList(head);
71
72      /* Let us delete the first node */
73      printf("\nDeleting first node ");
74      deleteNode(head, head);
75
76      printf("\nModified Linked List: ");
77      printList(head);
78
79      getchar();
80      return 0;
81  }
```

3. Presentation of Results

1.1) and 1.3)

```
Enter the total number of nodes:
3
Enter the data of node 1: 1
Enter the data of node 2: 2
Enter the data of node 3: 3
SINGLY LINKED LIST CREATED SUCCESSFULLY

Data in the list
data of list is:
1 2 3
Enter data to insert at beginning of the list: 4
DATA INSERTED SUCCESSFULLY

Data in the list
data of list is:
4 1 2 3
RUN SUCCESSFUL (total time: 7s)
```

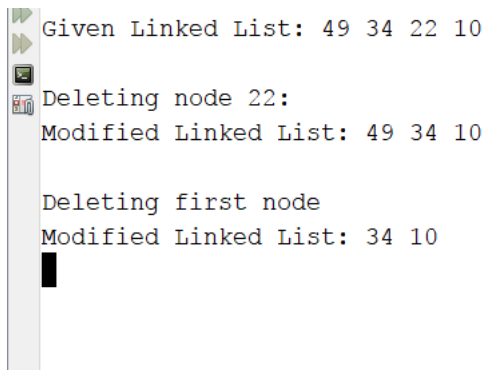
1.2)

```
Enter the total number of nodes: 2
Enter the data of node 1: 45
Enter the data of node 2: 56
SINGLY LINKED LIST CREATED SUCCESSFULLY

Data in the list
list is :-
45 56
Enter data to insert at end of the list: 78
DATA INSERTED SUCCESSFULLY

Data in the list
list is :-
45 56 78
RUN SUCCESSFUL (total time: 12s)
```

Q2

A screenshot of a terminal window with a light gray background. On the left side, there is a vertical toolbar with several icons: a green double arrow, a magnifying glass, a document with a pencil, a document with a trash can, and a document with a refresh symbol. The terminal text is as follows:

```
Given Linked List: 49 34 22 10  
Deleting node 22:  
Modified Linked List: 49 34 10  
  
Deleting first node  
Modified Linked List: 34 10  
█
```

4. Conclusions :

All the programs have been executed successfully.

Linked lists concepts have been revised.