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)

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

67

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

Roll Number: 18ETCS002147

```
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
   printf("Unable to allocate memory.");
76
77
78
          else
   79
          {
80
              newNode->data = data; // Link data part
              newNode->next = head; // Link address part
81
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;
         if(head == NULL)
92
93 🖹
94
              printf("List is empty.");
95
96
          else
97 🖨
98
              printf("data of list is: \n");
99
              temp = head;
              while(temp != NULL)
00
L01 🖹
L02
                  printf("%d ", temp->data); // Print data of current node
L03
                  temp = temp->next;
                                                // Move to next node
L04
L05
106
```

```
= #include <stdio.h>
   #include <stdlib.h>
  □ struct node {
 3
         int data;
 4
                            // Data
         struct node *next; // Address
 5
   }*head;
 6
7
     void createList(int n);
 8
     void insertNodeAtEnd(int data);
 9
     void displayList();
10
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");
         displayList();
20
         printf("\nEnter data to insert at end of the list: ");
21
22
         scanf("%d", &data);
23
         insertNodeAtEnd(data);
```

if(head == NULL)

```
24
          printf("\nData in the list \n");
25
26
          displayList();
27
         return 0;
28
29
30
31
   - /*
     * Create a list of n nodes
32
33
     void createList(int n)
34
35
          struct node *newNode, *temp;
36
37
          int data, i;
38
          head = (struct node *)malloc(sizeof(struct node));
39
40
41
          * If unable to allocate memory for head node
42
43
```

Roll Number: 18ETCS002147

```
47
          }
48
          else
49
          {
50
               * Reads data of node from the user
51
52
              printf("Enter the data of node 1: ");
53
54
              scanf("%d", &data);
55
56
              head->data = data; // Link the data field with data
              head->next = NULL; // Link the address field to NULL
57
58
59
              temp = head;
60
   白
61
62
               * Create n nodes and adds to linked list
63
              for(i=2; i<=n; i++)</pre>
64
65
                  newNode = (struct node *)malloc(sizeof(struct node));
66
67
```

printf("Unable to allocate memory.");

Name: KAUSHAL VASHISTH

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

Roll Number: 18ETCS002147

```
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.");
00
101
          else
102
L03
              newNode->data = data; // Link the data part
L04
              newNode->next = NULL;
L05
106
              temp = head;
L07
108
              // Traverse to the last node
L09
              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
       * Display entire list
119
120
121
       void displayList()
122
123
            struct node *temp;
124
125
            /*
            * If the list is empty i.e. head = NULL
126
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
                 printf(" %d ", temp->data); // Print data of current node
138
139
                 temp = temp->next;
                                                 // Move to next node
140
141
142
```

Roll Number: 18ETCS002147

Activate Winds

1.3)

```
#include <stdio.h>
#include <stdib.h>
struct Node

int data;
struct Node *next;

;;
```

```
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
             }
             head->data = head->next->data;
16
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
```

56

```
/* Utility function to insert a node at the beginning */
36
     void push(struct Node **head ref, int new data)
37
  □ {
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
     /* Utility function to print a linked list */
46
     void printList(struct Node *head)
47
48
   □ {
         while (head!=NULL)
49
   50
             printf("%d ",head->data);
51
52
             head=head->next;
53
54
         printf("\n");
55
```

Roll Number: 18ETCS002147

```
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
          printf("Given Linked List: ");
64
65
          printList(head);
          printf("\nDeleting node %d: ", head->next->next->data);
66
          deleteNode(head, head->next->next);
67
68
69
          printf("\nModified Linked List: ");
70
          printList(head);
71
72
          /* Let us delete the first node */
          printf("\nDeleting first node ");
73
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)

Name: KAUSHAL VASHISTH

```
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)
```

Roll Number: 18ETCS002147

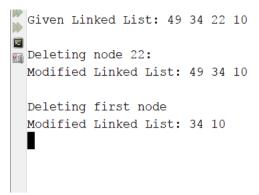
1.2)

```
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



4. Conclusions:

All the programs have been executed successfully.

Linked lists concepts have been revised.