

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 struct node {
4     int data;
5     struct node *next;
6 };
7 struct node *head = NULL;
8
9 void createList(int n) {
10 void insertAtBeginning(int data) {
11     struct node *newNode = (struct node*)malloc(sizeof(struct node));
12     if (newNode == NULL) {
13         printf("Memory allocation failed.\n");
14         return;
15     }
16     newNode->data = data;
17     newNode->next = head;
18     head = newNode;
19     printf("Node inserted at the beginning\n");
20 }
21 void insertAtEnd(int data) {
22     struct node *newNode = (struct node*)malloc(sizeof(struct node));
23     if (newNode == NULL) {
24         printf("Memory allocation failed.\n");
25         return;
26     }
27     newNode->data = data;
28     newNode->next = NULL;
29
30     if (head == NULL) {
31         head = newNode;
32     }
33     else {
34         struct node *temp = head;
35         while (temp->next != NULL)
36             temp = temp->next;
37         temp->next = newNode;
38     }
39     printf("Node inserted at the end\n");
40 }
41 void insertAtPosition(int data, int pos) {
42     int i;
43     struct node *newNode, *temp = head;
44
45     if (pos < 1) {
46         printf("Invalid position. Position must be 1 or greater.\n");
47         return;
48     }
49     if (pos == 1) {
50         insertAtBeginning(data);
51         return;
52     }
53     for (i = 1; i < pos - 1 && temp != NULL; i++)
54         temp = temp->next;
55     if (temp == NULL) {
56         printf("Position out of range: List is not long enough to reach position %d.\n", pos);
57         return;
58     }
59     newNode = (struct node*)malloc(sizeof(struct node));
60     if (newNode == NULL) {
61         printf("Memory allocation failed.\n");
62         return;
63     }
64     newNode->data = data;
65     newNode->next = temp->next;
66     temp->next = newNode;
67     printf("Node inserted at position %d\n", pos);
68 }
69
70 void displayList() {
71     struct node *temp = head;
72     if (head == NULL) {
73         printf("List is empty\n");
74         return;
75     }
76     printf("\nLinked list: ");
77     while (temp != NULL) {
78         printf("%d -> ", temp->data);
79         temp = temp->next;
80     }
81     printf("NULL\n");
82 }
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113

```

```
114 int main() {
115     int choice, n, data, pos;
116
117     while (1) {
118         printf("\n---- Singly Linked List Operations ----\n");
119         printf("1. Create linked list\n");
120         printf("2. Insert at Beginning\n");
121         printf("3. Insert at any Position\n");
122         printf("4. Insert at End\n");
123         printf("5. Display list\n");
124         printf("6. Exit\n");
125         printf("Enter your choice: ");
126         if (scanf("%d", &choice) != 1) {
127
128             while (getchar() != '\n');
129             printf("Invalid input. Please enter a number.\n");
130             continue;
131         }
132
133         switch (choice) {
134             case 1:
135                 printf("Enter number of nodes: ");
136                 scanf("%d", &n);
137                 createList(n);
138                 break;
139             case 2:
140                 printf("Enter data to insert: ");
141                 scanf("%d", &data);
142                 insertAtBeginning(data);
143                 break;
144             case 3:
145                 printf("Enter data: ");
146                 scanf("%d", &data);
147                 printf("Enter position: ");
148                 scanf("%d", &pos);
149                 insertAtPosition(data, pos);
150                 break;
151             case 4:
152                 printf("Enter data to insert: ");
153                 scanf("%d", &data);
154                 insertAtEnd(data);
155                 break;
156             case 5:
157                 displayList();
158                 break;
159             case 6:
160                 printf("Exiting...\n");
161                 exit(0);
162             default:
163                 printf("Invalid choice. Try again.\n");
164         }
165     }
166     return 0;

```

```
---- Singly Linked List Operations ----
1. Create linked list
2. Insert at Beginning
3. Insert at any Position
4. Insert at End
5. Display list
6. Exit
Enter your choice: 1
Enter number of nodes: 3
Enter data for node 1: 10
Enter data for node 2: 20
Enter data for node 3: 30

Linked list created successfully

---- Singly Linked List Operations ----
1. Create linked list
2. Insert at Beginning
3. Insert at any Position
4. Insert at End
5. Display list
6. Exit
Enter your choice: 2
Enter data to insert: 10
Node inserted at the beginning

---- Singly Linked List Operations ----
1. Create linked list
2. Insert at Beginning
3. Insert at any Position
4. Insert at End
5. Display list
6. Exit
Enter your choice: 3
Enter data: 4
Enter position: 3
Node inserted at position 3

---- Singly Linked List Operations ----
1. Create linked list
2. Insert at Beginning
3. Insert at any Position
4. Insert at End
5. Display list
6. Exit
Enter your choice: 4
Enter data to insert: 6
Node inserted at the end

---- Singly Linked List Operations ----
1. Create linked list
2. Insert at Beginning
3. Insert at any Position
4. Insert at End
5. Display list
6. Exit
Enter your choice: 6
Exiting...

Process returned 0 (0x0)  execution time : 41.343 s
Press any key to continue.
```

Lab Program-4

Q) WAP to implement single linked list with the following operations

- Create a Linked List
- Insertion of a node at
  - first position
  - any position
  - End position
- Display the contents of linked lists

pseudo code ~

```
BEGIN
    Define Structure Node with .contents data, next pointer
    Declare head pointer to NULL
    Function createLinkedList(n)
        Declare i, Data, temp, newnode
        For i ← 1 To n Do
            print "Enter value"
            Read value from Input
            newnode ← (structNode*) malloc(sizeof(structNode))
            newnode.data ← data
            newnode.next ← NULL
        If head == NULL
            head ← temp ← newnode
        Else
            temp.next ← newnode
        END IF
    END Function
```

```
Function InsertAtBeginning(Data)
    newnode ← (structNode*) malloc(sizeof(structNode))
    newnode.data ← data
    newnode.next ← head
    head ← newnode
```

End Function.

```

void InsertAtPosition (char data, int pos) {
    struct Node *temp = head;
    if (pos < 0) {
        printf ("Invalid position\n");
        return;
    }
    if (pos == 1) {
        insertBeginning (data);
        return;
    }
    for (int i = 1; i < pos - 1; i++) {
        temp = temp->next;
    }
    if (temp == NULL) {
        printf ("Position out of range\n");
        return;
    }
    struct Node *newnode = (struct Node *) malloc (sizeof (struct Node));
    newnode->data = data;
    newnode->next = temp->next;
    temp->next = newnode;
    printf ("Node inserted at position %d in %s", pos);
}

void displayList () {
    struct Node *temp = head;
    if (head == NULL) {
        printf ("List is Empty\n");
        return;
    }
    while (temp != NULL) {
        printf ("%d ", temp->data);
        temp = temp->next;
    }
    printf ("\n");
}

```

```

int main() {
    int choice, n, data, pos;
    char ch;
    printf ("In ... Simple Linked List operations ... \n");
    printf ("1. Create linked list\n");
    printf ("2. Insert at beginning\n");
    printf ("3. Insert at any position\n");
    printf ("4. Display List\n");
    printf ("5. Enter your choice : ");
    if (scanf ("%d", &choice) != 1) {
        while (getchar() != '\n');
        printf ("Invalid input, please enter a number\n");
        continue;
    }
    switch (choice) {
        case 1:
            printf ("Enter numbers & nodes : ");
            scanf ("%d", &n);
            createlist (&n);
            break;
        case 2:
            printf ("Enter data & Pos : ");
            scanf ("%d", &data);
            insertAtBeginning (data);
            break;
        case 3:
            printf ("Enter data : ");
            scanf ("%d", &data);
            printf ("Enter Position : ");
            scanf ("%d", &pos);
            insertAtPosition (data, pos);
            break;
        case 4:
            printf ("Enter data to insert : ");
            scanf ("%d", &data);
            insertAtEnd (data);
            break;
        case 5:
            displaylist();
            break;
        default:
            printf ("Ending... \n");
            exit(0);
    }
    printf ("Invalid choice, try again\n");
}

```

```

function InsertAtEnd (data)
    newnode = allocate memory
    newnode.data = data
    newnode.next = null
    if (head == null)
        head = newnode
    else
        struct Node *temp
        while (temp->next != null)
            temp = temp->next
        temp.next = newnode
end function

function insertAtAnyPosition (data, pos)
    newnode = allocate memory
    newnode.data = data
    if (pos == 1)
        newnode.next = head
        head = newnode
    else
        struct Node *temp
        count = 1
        while (temp.next != null And count < pos-1)
            temp = temp->next
            count++
        end while
        if (temp.next == null)
            temp.next = newnode
            newnode.next = null
        else
            newnode.next = temp.next
            temp.next = newnode
    end function

```

```

Code 1:
#include < stdio.h >
struct Node {
    int data;
    struct Node *next;
};

struct Node *head = NULL;

void createlist (int n) {
    struct Node *newnode, *temp;
    for (int i = 1; i < n; i++) {
        newnode = (struct Node *) malloc (sizeof (struct Node));
        newnode->data = value;
        newnode->next = NULL;
        if (head == NULL) {
            head = newnode;
            temp = newnode;
        } else {
            temp->next = newnode;
            temp = newnode;
        }
    }
}

void insertAtBeginning (int data) {
    struct Node *newnode = (struct Node *) malloc (sizeof (struct Node));
    if (newnode == NULL) {
        printf ("Memory allocation failed\n");
        return;
    }
    newnode->data = data;
    newnode->next = head;
    head = newnode;
    printf ("Node inserted at the beginning\n");
}

void insertAtEnd (int data) {
    struct Node *newnode = (struct Node *) malloc (sizeof (struct Node));
    if (newnode == NULL) {
        printf ("Memory allocation failed\n");
        return;
    }
    newnode->data = data;
    newnode->next = NULL;
    if (head == NULL) {
        head = newnode;
    } else {
        struct Node *temp = head;
        while (temp->next != NULL)
            temp = temp->next;
        temp->next = newnode;
    }
}
```

Lab 5  
@ Go imp  
LL & LR  
pseudo code

----- Singly Linked List Operations -----

1. Create linked list
2. Insert at Beginning
3. Insert at any position
4. Insert at End
5. Display List
6. Exit

Enter your choice : 1  
Enter number of nodes : 3  
Enter data for node 1: 10  
Enter data for node 2: 20  
Enter data for node 3: 30  
Linked List created successfully.

----- Singly Linked List Operations -----

1. Create linked list
2. Insert at Beginning
3. Insert at any position
4. Insert at End
5. Display List
6. Exit

Enter your choice : 2  
Enter data to insert : 10  
Node inserted at the beginning.

----- Singly Linked List Operations -----

1. Create linked list
2. Insert at Beginning
3. Insert at any position
4. Insert at End
5. Display List
6. Exit

Enter your choice : 3  
Enter data : 4  
Enter position: 3  
Node inserted at position 3

Enter your choice : 4  
Enter data to insert : 6  
Node inserted at the End

Enter your choice : 1  
Exiting

~~Lab 5~~