

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4
5 struct Node {
6     int data;
7     struct Node *next;
8 };
9
10 struct Node *head = NULL;
11
12 struct Node* createNode(int data) {
13     struct Node *newNode = (struct Node*)malloc(sizeof(struct Node));
14     if (newNode == NULL) {
15         printf("Memory allocation failed\n");
16         exit(1);
17     }
18     newNode->data = data;
19     newNode->next = NULL;
20     return newNode;
21 }
22
23
24 void insertAtBeginning(int data) {
25     struct Node *newNode = createNode(data);
26     newNode->next = head;
27     head = newNode;
28     printf("Node inserted at the beginning.\n");
29 }
30
31
32 void insertAtEnd(int data) {
33     struct Node *newNode = createNode(data);
34     if (head == NULL) {
35         head = newNode;
36     } else {
37         struct Node *temp = head;
38         while (temp->next != NULL)
39             temp = temp->next;
40         temp->next = newNode;
41     }
42     printf("Node inserted at the end.\n");
43 }
44
45 void insertAtPosition(int data, int position) {
46     int i;
47     struct Node *newNode = createNode(data);
48
49     if (position == 1) {
50         newNode->next = head;
51         head = newNode;
52         printf("Node inserted at position 1.\n");
53         return;
54     }
55
56     struct Node *temp = head;
57     for (i = 1; i < position - 1 && temp != NULL; i++) {
58         temp = temp->next;
59     }
60
61     if (temp == NULL) {
62         printf("Position out of range!\n");
63         free(newNode);
64         return;
65     }
66
67     newNode->next = temp->next;
68     temp->next = newNode;
69     printf("Node inserted at position %d.\n", position);
70 }
71
72
73 void displayList() {
74     if (head == NULL) {

```

```
73 void displayList() {
74     if (head == NULL) {
75         printf("List is empty.\n");
76         return;
77     }
78
79     struct Node *temp = head;
80     printf("Linked List: ");
81     while (temp != NULL) {
82         printf("%d -> ", temp->data);
83         temp = temp->next;
84     }
85     printf("NULL\n");
86 }
87
88
89 int main() {
90     int choice, data, position;
91
92     while (1) {
93         printf("\n Singly Linked List Menu\n");
94         printf("1. Insert at Beginning\n");
95         printf("2. Insert at Position\n");
96         printf("3. Insert at End\n");
97         printf("4. Display List\n");
98         printf("5. Exit\n");
99         printf("Enter your choice: ");
100        scanf("%d", &choice);
101
102        switch (choice) {
103            case 1:
104                printf("Enter data to insert: ");
105                scanf("%d", &data);
106                insertAtBeginning(data);
107                break;
108            case 2:
109                printf("Enter data to insert: ");
110                scanf("%d", &data);
```

```
    scanf("%d", &data);
    insertAtBeginning(data);
    break;
case 2:
    printf("Enter data to insert: ");
    scanf("%d", &data);
    printf("Enter position: ");
    scanf("%d", &position);
    insertAtPosition(data, position);
    break;
case 3:
    printf("Enter data to insert: ");
    scanf("%d", &data);
    insertAtEnd(data);
    break;
case 4:
    displayList();
    break;
case 5:
    printf("Exiting program.\n");
    exit(0);
default:
    printf("Invalid choice! Try again.\n");
}
}

return 0;
}
```

C:\ "C:\Users\BMSCE\Documents" X + | v

```
Enter your choice: 1
Enter data to insert: 9
Node inserted at the beginning.
```

```
Singly Linked List Menu
1. Insert at Beginning
2. Insert at Position
3. Insert at End
4. Display List
5. Exit
```

```
Enter your choice: 1
Enter data to insert: 8
Node inserted at the beginning.
```

```
Singly Linked List Menu
1. Insert at Beginning
2. Insert at Position
3. Insert at End
4. Display List
5. Exit
```

```
Enter your choice: 4
Linked List: 8 -> 9 -> 5 -> NULL
```

```
Singly Linked List Menu
1. Insert at Beginning
2. Insert at Position
3. Insert at End
4. Display List
5. Exit
```

```
Enter your choice: |
```