

## Experiments\SinglyLinkedList.c

```
1  #include<stdio.h>
2  #include<malloc.h>
3  struct Node
4  {
5      int data;
6      struct Node *next;
7  };
8  // Function to create a new node
9  struct Node *createNode(int data) {
10     struct Node *newNode = (struct Node*)malloc(sizeof(struct Node));
11     newNode->data = data;
12     newNode->next = NULL;
13     return newNode;
14 }
15 // Function to insert a new element at the beginning of the singly linked list
16 void insertAtFirst(struct Node* *start, int data) {
17     struct Node* newNode = createNode(data);
18     newNode->next = *start;
19     *start = newNode;
20 }
21 // Function to insert a new element at the end of the singly linked list
22 void insertAtEnd(struct Node** head, int data) {
23     struct Node* newNode = createNode(data);
24     if (*head == NULL) {
25         *head = newNode;
26         return;
27     }
28     struct Node* temp = *head;
29     while (temp->next != NULL) {
30         temp = temp->next;
31     }
32     temp->next = newNode;
33 }
34 // Function to insert a new element at a specific position in the singly linked list
35 void insertAtPosition(struct Node** head, int data, int position) {
36     struct Node* newNode = createNode(data);
37     if (position == 0) {
38         insertAtFirst(head,data);
39         return;
40     }
41     struct Node* temp = *head;
42     for (int i = 0; temp != NULL && i < position - 1; i++) {
43         temp = temp->next;
44     }
45     if (temp == NULL) {
46         printf("Position out of range\n");
47         free(newNode);
48         return;
49     }
```

```
49     }
50     newNode->next = temp->next;
51     temp->next = newNode;
52 }
53 // Function to delete the first node of the singly linked list
54 void deleteFromFirst(struct Node** head) {
55     if (*head == NULL) {
56         printf("List is empty\n");
57         return;
58     }
59     struct Node* temp = *head;
60     *head = temp->next;
61     free(temp);
62 }
63 // Function to delete the last node of the singly linked list
64 void deleteFromEnd(struct Node** head) {
65     if (*head == NULL) {
66         printf("List is empty\n");
67         return;
68     }
69     struct Node* temp = *head;
70     if (temp->next == NULL) {
71         free(temp);
72         *head = NULL;
73         return;
74     }
75     while (temp->next->next != NULL) {
76         temp = temp->next;
77     }
78     free(temp->next);
79     temp->next = NULL;
80 }
81 // Function to delete a node at a specific position in the singly linked list
82 void deleteAtPosition(struct Node** head, int position) {
83     if (*head == NULL) {
84         printf("List is empty\n");
85         return;
86     }
87     struct Node* temp = *head;
88     if (position == 0) {
89         deleteFromFirst(head);
90         return;
91     }
92     for (int i = 0; temp != NULL && i < position - 1; i++) {
93         temp = temp->next;
94     }
95     if (temp == NULL || temp->next == NULL) {
96         printf("Position out of range\n");
97         return;
98     }
```

```
99  struct Node* next = temp->next->next;
100  free(temp->next);
101  temp->next = next;
102  }
103  // Function to print the LinkedList
104  void print(struct Node* head) {
105      struct Node* temp = head;
106      while (temp != NULL) {
107          printf("%d -> ", temp->data);
108          temp = temp->next;
109      }
110      printf("NULL\n");
111  }
112  // Driver Code
113  int main() {
114      struct Node* head = NULL;
115
116      insertAtFirst(&head, 10);
117      printf("Linked list after inserting the node:10 at the beginning \n");
118      print(head);
119
120      printf("Linked list after inserting the node:20 at the end \n");
121      insertAtEnd(&head, 20);
122      print(head);
123
124      printf("Linked list after inserting the node:5 at the end \n");
125      insertAtEnd(&head, 5);
126      print(head);
127
128      printf("Linked list after inserting the node:30 at the end \n");
129      insertAtEnd(&head, 30);
130      print(head);
131
132      printf("Linked list after inserting the node:15 at position 2 \n");
133      insertAtPosition(&head, 15, 2);
134      print(head);
135
136      printf("Linked list after deleting the first node: \n");
137      deleteFromFirst(&head);
138      print(head);
139
140      printf("Linked list after deleting the last node: \n");
141      deleteFromEnd(&head);
142      print(head);
143
144      printf("Linked list after deleting the node at position 1: \n");
145      deleteAtPosition(&head, 1);
146      print(head);
147      return 0;
148  }
```