```c
#include<stdio.h>
#include<stdlib.h>

struct Node{
    int data;
    struct Node * next;
};

void linkedListTraversal(struct Node *ptr)
{
    while (ptr != NULL)
    {
        printf("Element: %d\n", ptr->data);
        ptr = ptr->next;
    }
}

// Case 1
struct Node * insertAtFirst(struct Node *head, int data){
    struct Node * ptr = (struct Node *) malloc(sizeof(struct Node
    ptr->data = data;

    ptr->next = head;
    return ptr;
}

// Case 2
struct Node * insertAtIndex(struct Node *head, int data, int inde
    struct Node * ptr = (struct Node *) malloc(sizeof(struct Node
    struct Node * p = head;
    int i = 0;

    while (i!=index-1)
    {
        p = p->next;
        i++;
    }
    ptr->data = data;
    ptr->next = p->next;
```

```
40:     p->next = ptr;
41:     return head;
42: }
43:
44: // Case 3
45: struct Node * insertAtEnd(struct Node *head, int data){
46:     struct Node * ptr = (struct Node *) malloc(sizeof(struct Node)
47:     ptr->data = data;
48:     struct Node * p = head;
49:
50:     while(p->next!=NULL){
51:         p = p->next;
52:     }
53:     p->next = ptr;
54:     ptr->next = NULL;
55:     return head;
56: }
57:
58: // Case 4
59: struct Node * insertAfterNode(struct Node *head, struct Node *prev
60:     struct Node * ptr = (struct Node *) malloc(sizeof(struct Node)
61:     ptr->data = data;
62:
63:     ptr->next = prevNode->next;
64:     prevNode->next = ptr;
65:
66:
67:     return head;
68: }
69:
70:
71: int main(){
72:     struct Node *head;
73:     struct Node *second;
74:     struct Node *third;
75:     struct Node *fourth;
76:
77:     // Allocate memory for nodes in the linked list in Heap
78:     head = (struct Node *)malloc(sizeof(struct Node));
```

```
79:     second = (struct Node *)malloc(sizeof(struct Node));
80:     third = (struct Node *)malloc(sizeof(struct Node));
81:     fourth = (struct Node *)malloc(sizeof(struct Node));
82:
83:     // Link first and second nodes
84:     head->data = 7;
85:     head->next = second;
86:
87:     // Link second and third nodes
88:     second->data = 11;
89:     second->next = third;
90:
91:     // Link third and fourth nodes
92:     third->data = 41;
93:     third->next = fourth;
94:
95:     // Terminate the list at the third node
96:     fourth->data = 66;
97:     fourth->next = NULL;
98:
99:     printf("Linked list before insertion\n");
100:    linkedListTraversal(head);
101:
102:     printf("\nLinked list after insertion at beginning\n");
103:    head = insertAtFirst(head, 39);
104:    linkedListTraversal(head);
105:
106:    printf("\nLinked list after insertion at an index\n");
107:    head = insertAtIndex(head, 73, 1);
108:    linkedListTraversal(head);
109:
110:    printf("\nLinked list after insertion at the end\n");
111:    head = insertAtEnd(head, 99);
112:    linkedListTraversal(head);
113:
114:    printf("\nLinked list after insertion after a node\n");
115:    head = insertAfterNode(head, third, 45);
116:    linkedListTraversal(head);
117:
```

```
118:
119:     return 0;
120: }
121:
122:
123:
124:
125:
```