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

// Defining the node structure
struct node {
   int data;
   struct node *next;
};

// Function to display and print the linked list
void display(struct node *ptr) {
   printf("\nLinked List: ");
   if (ptr == NULL) {
      printf("List is empty.\n");
   } else {
      while (ptr != NULL) {
         printf("-> %d ", ptr->data);
         ptr = ptr->next;
      }
      printf("\n");
   }
}

// Function to insert a node at the beginning of the list
struct node *insertatbeg(struct node *head, int data) {
   struct node *ptr = (struct node *)malloc(sizeof(struct node));
   ptr->data = data;
   ptr->next = head;
   return ptr;
}
```

```
// Function to insert a node at a given index
struct node *insertatindex(struct node *head, int data, int index) {
    struct node *ptr = (struct node *)malloc(sizeof(struct node));
    struct node *p = head;
    int i = 0;

    // Handle insertion at the beginning separately
    if (index == 0) {
        return insertatbeg(head, data);
    }

    // Traverse the list to reach the desired position
    while (i < index - 1 && p != NULL) {
        p = p->next;
        i++;
    }

    // Check if the index is out of bounds
    if (p == NULL) {
        printf("\nIndex out of bounds.\n");
        return head;
    }

    ptr->data = data;
    ptr->next = p->next;
    p->next = ptr;
    return head;
}

// Function to insert a node at the end of the list
struct node *insertatend(struct node *head, int data) {
```

```
    struct node *ptr = (struct node *)malloc(sizeof(struct node));
    struct node *p = head;

    ptr->data = data;
    ptr->next = NULL;

    // If the list is empty, the new node becomes the head
    if (head == NULL) {
        return ptr;
    }

    // Traverse to the end of the list
    while (p->next != NULL) {
        p = p->next;
    }
    p->next = ptr;
    return head;
}

// Function to delete the node at the beginning of the list
struct node *deleteatbeg(struct node *head) {
    if (head == NULL) {
        printf("\nList is empty.\n");
        return NULL;
    }
    struct node *ptr = head;
    head = head->next;
    free(ptr);
    return head;
}
```

```
// Function to delete the node at a given index
struct node *deleteatindex(struct node *head, int index) {
    if (head == NULL) {
        printf("\nList is empty.\n");
        return NULL;
    }
    struct node *p = head;
    struct node *q = head->next;

    // Handle deletion at the beginning separately
    if (index == 0) {
        return deleteatbeg(head);
    }

    // Traverse to the node before the node to be deleted
    for (int i = 0; i < index - 1; i++) {
        if (p->next == NULL) {
            printf("\nIndex out of bounds.\n");
            return head;
        }
        p = p->next;
        q = q->next;
    }

    if (q == NULL) {
        printf("\nIndex out of bounds.\n");
        return head;
    }

    p->next = q->next;
    free(q);
```

```
    return head;
}

// Function to delete the node at the end of the list
struct node *deleteatend(struct node *head) {
    if (head == NULL) {
        printf("\nList is empty.\n");
        return NULL;
    }

    struct node *p = head;
    struct node *q = head->next;

    // Handle the case with only one element in the list
    if (q == NULL) {
        free(p);
        return NULL;
    }

    // Traverse to the second last element
    while (q->next != NULL) {
        p = p->next;
        q = q->next;
    }

    p->next = NULL;
    free(q);
    return head;
}

int main() {
```

```
struct node *head = NULL; // Initialize head as NULL to signify an empty list
int data, ch, index;

while (1) {
   printf("\n\nMenu:\n");
   printf("1. Insert node at front\n");
   printf("2. Insert node at Index\n");
   printf("3. Insert node at end\n");
   printf("4. Delete node at front\n");
   printf("5. Delete node at index\n");
   printf("6. Delete node at end\n");
   printf("7. Display nodes\n"); // Added display option
   printf("0. Exit\n");
   printf("Enter your choice: ");
   scanf("%d", &ch);

   switch (ch) {
      case 1:
         printf("\nEnter the data to insert at the beginning: ");
         scanf("%d", &data);
         head = insertatbeg(head, data);
         display(head);
         break;

      case 2:
         printf("\nEnter the data to insert at the given index: ");
         scanf("%d", &data);
         printf("Enter the index at which to insert: ");
         scanf("%d", &index);
         head = insertatindex(head, data, index);
         display(head);
```

```
    break;

case 3:
    printf("\nEnter the data to insert at the end: ");
    scanf("%d", &data);
    head = insertatend(head, data);
    display(head);
    break;

case 4:
    head = deleteatbeg(head);
    display(head);
    break;

case 5:
    printf("\nEnter the index at which to delete the node: ");
    scanf("%d", &index);
    head = deleteatindex(head, index);
    display(head);
    break;

case 6:
    head = deleteatend(head);
    display(head);
    break;

case 7: // New case for displaying the list
    display(head);
    break;

case 0:
```

```
            printf("\nThank you!\n");
             printf("\n Course Teacher : Mrs.Archana Chitte");
            exit(0);

        default:
            printf("\nInvalid choice. Please enter a valid option.\n");
      }
   }

   return 0;
}
```

```
Menu:
1. Insert node at front
2. Insert node at Index
3. Insert node at end
4. Delete node at front
5. Delete node at index
6. Delete node at end
7. Display nodes
0. Exit
Enter your choice: 1

Enter the data to insert at the beginning: 50

Linked List: -> 50


Menu:
1. Insert node at front
2. Insert node at Index
3. Insert node at end
4. Delete node at front
5. Delete node at index
6. Delete node at end
7. Display nodes
0. Exit
Enter your choice: 2

Enter the data to insert at the given index: 60
Enter the index at which to insert: 1

Linked List: -> 50 -> 60


Menu:
1. Insert node at front
2. Insert node at Index
3. Insert node at end
4. Delete node at front
5. Delete node at index
6. Delete node at end
7. Display nodes
```

```
Enter your choice: 3

Enter the data to insert at the end: 70

Linked List: -> 50 -> 60 -> 70


Menu:
1. Insert node at front
2. Insert node at Index
3. Insert node at end
4. Delete node at front
5. Delete node at index
6. Delete node at end
7. Display nodes
0. Exit
Enter your choice: 3

Enter the data to insert at the end: 80

Linked List: -> 50 -> 60 -> 70 -> 80


Menu:
1. Insert node at front
2. Insert node at Index
3. Insert node at end
4. Delete node at front
5. Delete node at index
6. Delete node at end
7. Display nodes
0. Exit
Enter your choice: 7

Linked List: -> 50 -> 60 -> 70 -> 80
```

```
Menu:
1. Insert node at front
2. Insert node at Index
3. Insert node at end
4. Delete node at front
5. Delete node at index
6. Delete node at end
7. Display nodes
0. Exit
Enter your choice: 4

Linked List: -> 60 -> 70 -> 80


Menu:
1. Insert node at front
2. Insert node at Index
3. Insert node at end
4. Delete node at front
5. Delete node at index
6. Delete node at end
7. Display nodes
0. Exit
Enter your choice: 5

Enter the index at which to delete the node: 1

Linked List: -> 60 -> 80


Menu:
1. Insert node at front
2. Insert node at Index
3. Insert node at end
4. Delete node at front
5. Delete node at index
6. Delete node at end
7. Display nodes
0. Exit
Enter your choice: 6
```

```
Enter your choice: 6

Linked List: -> 60


Menu:
1. Insert node at front
2. Insert node at Index
3. Insert node at end
4. Delete node at front
5. Delete node at index
6. Delete node at end
7. Display nodes
0. Exit
Enter your choice: 0

Thank you!

 Course Teacher : Mrs.Archana Chitte
PS C:\Users\varad_0kfzvy3\AppData\Local\Temp>
```