

```

1 #include <cs50.h>
2 #include <csLIB.h>
3
4 struct node {
5     int data;
6     struct node *next;
7 }
8
9 struct node *head = NULL;
10
11
12 void createList(int n) {
13     struct node *newNode, *temp = NULL;
14     int data, i;
15
16     if (n <= 0) {
17         printf("Number of nodes should be greater than 0\n");
18         return;
19     }
20
21
22     if (head != NULL) {
23         printf("Warning: Overwriting existing list.\n");
24         head = NULL;
25     }
26
27     for (i = 1; i <= n; i++) {
28         newNode = (struct node*)malloc(sizeof(struct node));
29         if (newNode == NULL) {
30             printf("Memory allocation failed\n");
31             return;
32         }
33         printf("Enter data for node %d: ", i);
34         scanf("%d", &data);
35
36         newNode->data = data;
37         newNode->next = NULL;
38
39         if (head == NULL)
40             head = newNode;
41         else
42             temp->next = newNode;
43
44         temp = newNode;
45     }
46
47     printf("\nLinked list created successfully\n");
48 }
49
50 void insertAtBeginning(int data) {
51     struct node *newNode = (struct node*)malloc(sizeof(struct node));
52
53     if (newNode == NULL) {
54         printf("Memory allocation failed.\n");
55         return;
56     }
57     newNode->data = data;
58     newNode->next = head;
59     head = newNode;
60
61     printf("Node inserted at the beginning\n");
62 }
63
64 void insertAtEnd(int data) {
65     struct node *newNode = (struct node*)malloc(sizeof(struct node));
66
67     if (newNode == NULL) {
68         printf("Memory allocation failed.\n");
69         return;
70     }
71     newNode->data = data;
72     newNode->next = NULL;
73
74     if (head == NULL) {
75         head = newNode;
76     }
77     else {
78         struct node *temp = head;
79         while (temp->next != NULL)
80             temp = temp->next;
81         temp->next = newNode;
82     }
83
84     printf("Node inserted at the end\n");
85 }
86
87 void insertAtPosition(int data, int pos) {
88     int i;
89     struct node *newNode, *temp = head;
90
91     if (pos < 1) {
92         printf("Invalid position. Position must be 1 or greater.\n");
93         return;
94     }
95
96     if (pos == 1) {
97         insertAtBeginning(data);
98         return;
99     }
100
101     for (i = 1; i < pos - 1 && temp != NULL; i++)
102         temp = temp->next;
103
104     temp = temp->next;
105

```

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

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

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;
}

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;
    }
    printf("Node inserted at position %d\n", pos);
}

void insertAtPosition(int data, int pos) {
    struct node *newNode = (struct node *)malloc(sizeof(struct node));
    if (newNode == NULL) {
        printf("Memory allocation failed.\n");
        return;
    }
    newNode->data = data;
    newNode->next = head;
    if (pos == 1) {
        insertAtBeginning(data);
        return;
    }
    struct node *temp = head;
    for (int i = 1; i < pos - 1; i++) {
        temp = temp->next;
    }
    temp->next = newNode;
    newNode->next = temp->next;
    printf("Node inserted at position %d\n", pos);
}

void displayList() {
    struct node *temp = head;
    if (head == NULL) {
        printf("List is empty.\n");
        return;
    }
    printf("Linked list: ");
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\nNULL\n");
}

int main() {
    int choice, n, data, pos;
    while (1) {
        printf("\n---- Singly 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. Insert at End\n");
        printf("5. Display list\n");
        printf("6. Exit\n");
        printf("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 number of nodes: ");
                scanf("%d", &n);
                createList(n);
                break;
            case 2:
                printf("Enter data to insert: ");
                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;
            case 6:
                printf("Exiting...\n");
                exit(0);
            default:
                printf("Invalid choice. Try again.\n");
        }
    }
    return 0;
}
```

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

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

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;
}

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;
    }
}

void insertAtPosition(int data, int pos) {
    struct node *newNode = (struct node *)malloc(sizeof(struct node));
    if (newNode == NULL) {
        printf("Memory allocation failed.\n");
        return;
    }
    newNode->data = data;
    newNode->next = head;
    if (pos == 1) {
        insertAtBeginning(data);
        return;
    }
    struct node *temp = head;
    for (int i = 1; i < pos - 1; i++) {
        temp = temp->next;
    }
    temp->next = newNode;
    newNode->next = temp->next;
}

void displayList() {
    struct node *temp = head;
    if (head == NULL) {
        printf("List is empty.\n");
        return;
    }
    printf("Linked list: ");
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\nNULL\n");
}

int main() {
    int choice, n, data, pos;
    while (1) {
        printf("\n---- Singly 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. Insert at End\n");
        printf("5. Display list\n");
        printf("6. Exit\n");
        printf("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 number of nodes: ");
                scanf("%d", &n);
                createList(n);
                break;
            case 2:
                printf("Enter data to insert: ");
                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;
            case 6:
                printf("Exiting...\n");
                exit(0);
            default:
                printf("Invalid choice. Try again.\n");
        }
    }
    return 0;
}
```

```
D:\Dchethan\Documents\linkedlist.exe
---- 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: 2
Enter data for node 1: 3
Enter data for node 2: 4
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: 5
Linked list: 3 -> 4 -> NULL
---- 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: 344
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: 5
linked list: 344 -> 3 -> 4 -> NULL
---- 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: 678
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: 5
```

The screenshot shows a browser window with multiple tabs open. The active tab is 'Middle of the Linked List - LeetCode'. The page displays the solution code for this problem, which is a C function named `middleNode`. The code uses a fast & slow pointer method to find the middle node of a linked list. The code is as follows:

```
1 struct ListNode* middleNode(struct ListNode* head) {
2     struct ListNode* temp = head;
3
4     // Fast & Slow pointer method
5     while (temp != NULL && temp->next != NULL) {
6         head = head->next;
7         temp = temp->next->next;
8     }
9
10    return head;
11 }
12 }
```

The code editor shows the runtime statistics: 0 ms (Beats 100.00%) and 8.69 MB (Beats 12.13%). Below the code editor, there is a chart showing the distribution of execution times. The chart has a blue bar at 0 ms, with a tooltip indicating it is 100% of the time. There are also small bars for 1ms, 2ms, and 3ms.

DATE: 10/11/25 PAGE:

- 1) Write a program to implement singly linked list with  
the following  
a) Create a linked list  
b) Insertion at a node at  
first position \* any position \* end of the list  
c) Display the contents of linked list

pseudocode

structure node

data

    next → node

end structure

⇒ empty linked list = head ← null

⇒ New node creation

    createNode (value)

    newnode ← allocate memory for node

    newnode.data ← value

    newnode.next ← null

    return node

end function

⇒ Insert at first position

firstPosition (value)

    newnode ← createNode (value)

    newnode.next ← head

    head ← newnode

end function

⇒ Insert at end

insertAtEnd (value)

    newnode → createNode (value)

    if head = null then

        head ← newnode

    return



Shot on OnePlus

chE\_Reddy ← head

DATE:

while temp.next != null  
in loop do to until temp == temp.next  
end while  
temp.next = newnode  
end function

=> Insert at Any position

Insertatanyposition (targetvalue, newvalue)

temp = head

while temp != null and temp.data != targetvalue

temp = temp.next

end while

if temp = null then

print "target not found", 1, blank

return

newnode = createNode (newvalue)

newnode.next = temp.next

temp.next = newnode

end if

newnode = createNode (newvalue)

newnode.next = temp.next

temp.next = newnode

=> Display

Display ()

temp = head

while temp != null

print temp.data

temp = temp.next

end while

end function



Shot on OnePlus

chE\_Reddy

```

PAGE: DATE: PAGE:
#include <stdio.h>
#include <stdlib.h>

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

struct node* head = NULL;

void createlist(int n) {
    struct node* Newnode, * temp = NULL;
    int data, i;
    if (n <= 0) {
        printf("Number of nodes should be greater than 0\n");
        return;
    }
    if (head == NULL) {
        printf("Warning: Over-writing existing list.\n");
        head = Newnode;
    }
    for (i = 1; i <= n; i++) {
        Newnode = (struct node*) malloc(sizeof(struct node));
        if (Newnode == NULL) {
            printf("Memory allocation failed.\n");
            return;
        }
        printf("Enter data for node %d: ", i);
        scanf("%d", &data);
        Newnode->data = data;
        Newnode->next = NULL;
        if (head == NULL)
            head = Newnode;
        else
            temp->next = Newnode;
        temp = Newnode;
    }
}

Shot on OnePlus
chE_Reddy

```

DATE: \_\_\_\_\_

```

    printf("A linked list created\n");
}

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;
    }
    printf("Node inserted at the end\n");
}

```

Shot on OnePlus by chE\_Reddy

DATE: \_\_\_\_\_  
PAGE: \_\_\_\_\_

```

void insertatposition(int data, int pos) {
    int i;
    struct node * newnode, * temp = head;
    if (pos <= 0) {
        printf("Invalid position\n");
        return;
    }
    if (pos == 1) {
        insertAtBeginning(data);
        return;
    }
    for (i = 1; i < pos - 1 && temp != NULL; i++)
        temp = temp->next;
    if (temp == NULL) {
        printf("Position out of range\n", pos);
        return;
    }
    newnode = (struct node *) malloc(sizeof(struct node));
    if (newNode == NULL) {
        printf("Memory allocation failed\n");
        return;
    }
    newnode->data = data;
    newnode->next = temp->next;
    temp->next = newnode;
    printf("Node inserted at position\n", pos);
}

void displaylist() {
    struct node * temp = head;
    if (head == NULL) {
        printf("List is empty\n");
        return;
    }
}

```



Shot on OnePlus  
chE\_Reddy

```

printf("linked list:\n");
while (temp != NULL) {
    printf("%d ", temp->data);
    temp = temp->next;
}
printf("NULL\n");

int main() {
    int choice, n, data, pos;
    while (1) {
        printf("1. Create linked list\n");
        printf("2. Insert at Beginning\n");
        printf("3. Insert at Any position\n");
        printf("4. Insert at End\n");
        printf("5. Display list\n");
        printf("6. Exit\n");
        if (scanf("%d", &choice) != 1) {
            while (getchar() != '\n');
            printf("Invalid input. Please enter a number\n");
            continue;
        }
        switch (choice) {
            case 1:
                createList(n);
                break;
            case 2:
                printf("Enter number of nodes: ");
                scanf("%d", &n);
                createList(n);
                insertAtBeginning(data);
                break;
            case 3:
                printf("Enter data to insert: ");
                scanf("%d", &data);
                insertAtPosition(pos, data);
                break;
            case 4:
                printf("Enter data to insert: ");
                scanf("%d", &data);
                insertAtEnd(data);
                break;
            case 5:
                displayList();
                break;
            case 6:
                exit(0);
        }
    }
}

```

Shot on OnePlus  
chE\_Reddy

CASE3:

```
printf("Enter data:");  
scanf("%d", &data);  
printf("Enter position:");  
scanf("%d", &pos);  
insertatposition(data, pos);  
break;
```

CASE4:

```
printf("Enter data to insert");  
scanf("%d", &data);  
insertatend(data);  
break;
```

CASE5:

```
displaylist();  
break;
```

CASE6:

```
printf("Exiting");  
exit(0);
```

Output:

singly linked list operation.

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

Enter your choice:

Enter number of nodes: 3

Enter data for node 1: 10

Enter data for node 2: 20



Shot on OnePlus

chE\_Reddy

DATE:

Enter data for node 3: 30

Enter your choice is 5  
10 20 30

Enter your choice is 3  
Enter your data is 45  
10 20 30 45

Enter position is 3  
Node inserted at position 3

Enter your choice is 4  
Enter data is 50  
10 20 30 45 50

Node inserted at (ith) end  
10 20 30 45 50

~~10 20 30 45 50~~  
~~50 is inserted at the end~~



Shot on OnePlus  
chE\_Reddy