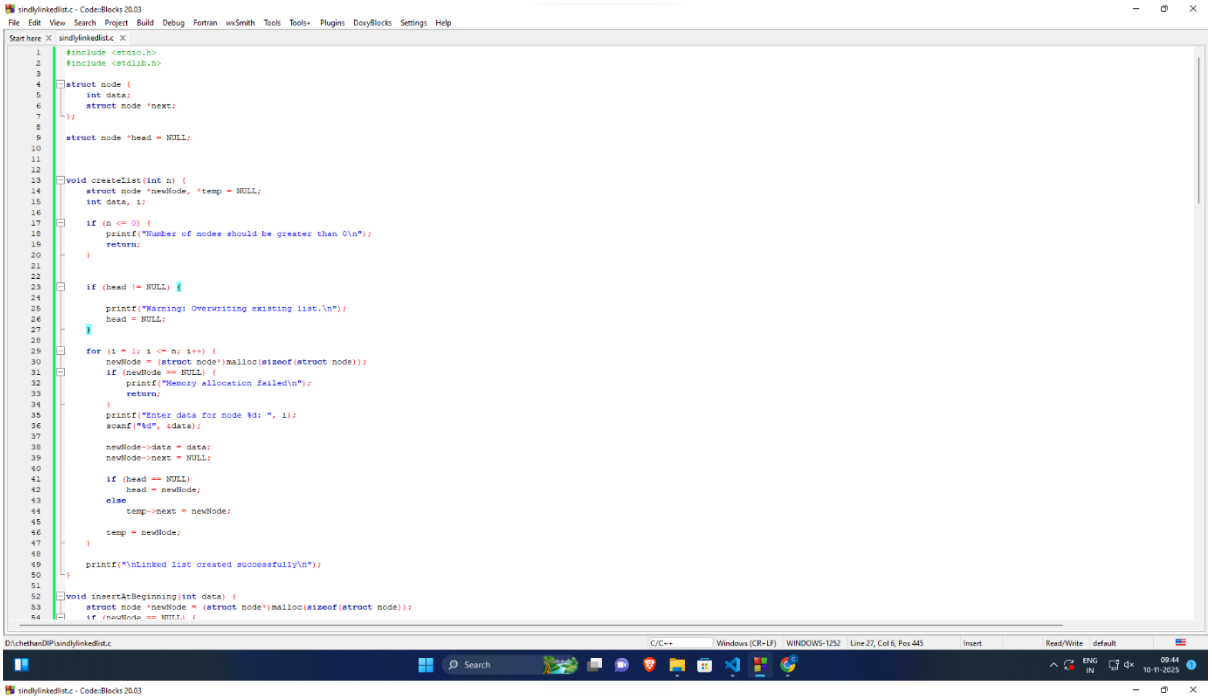
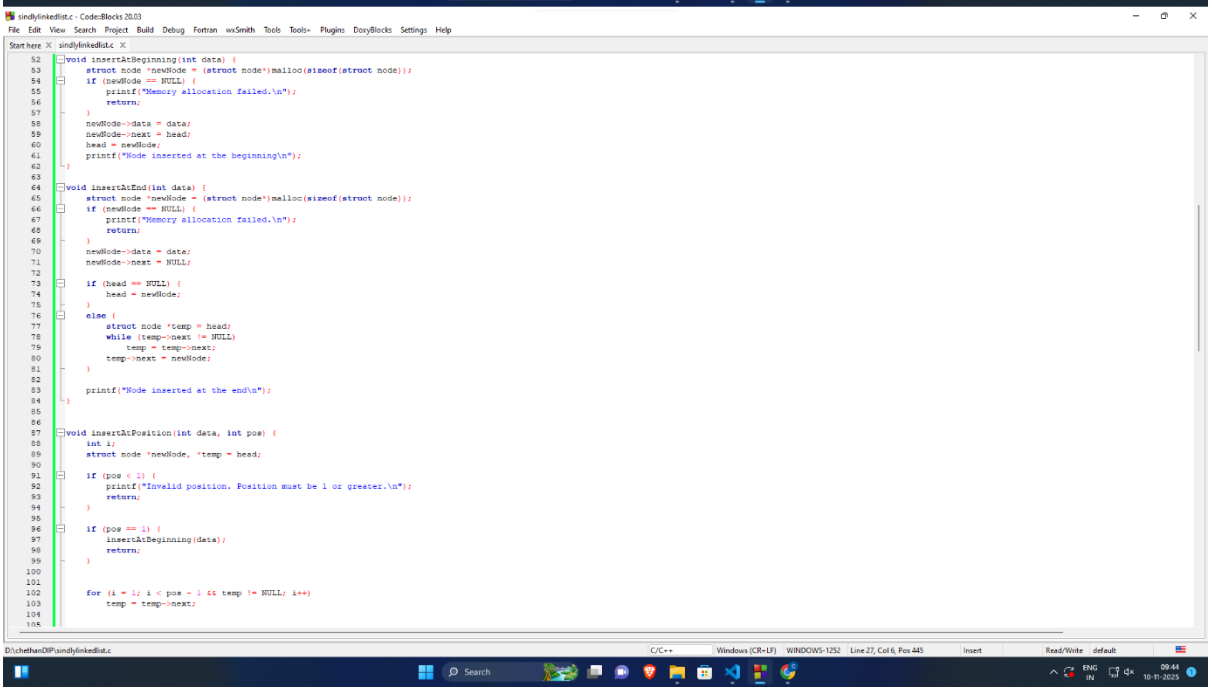
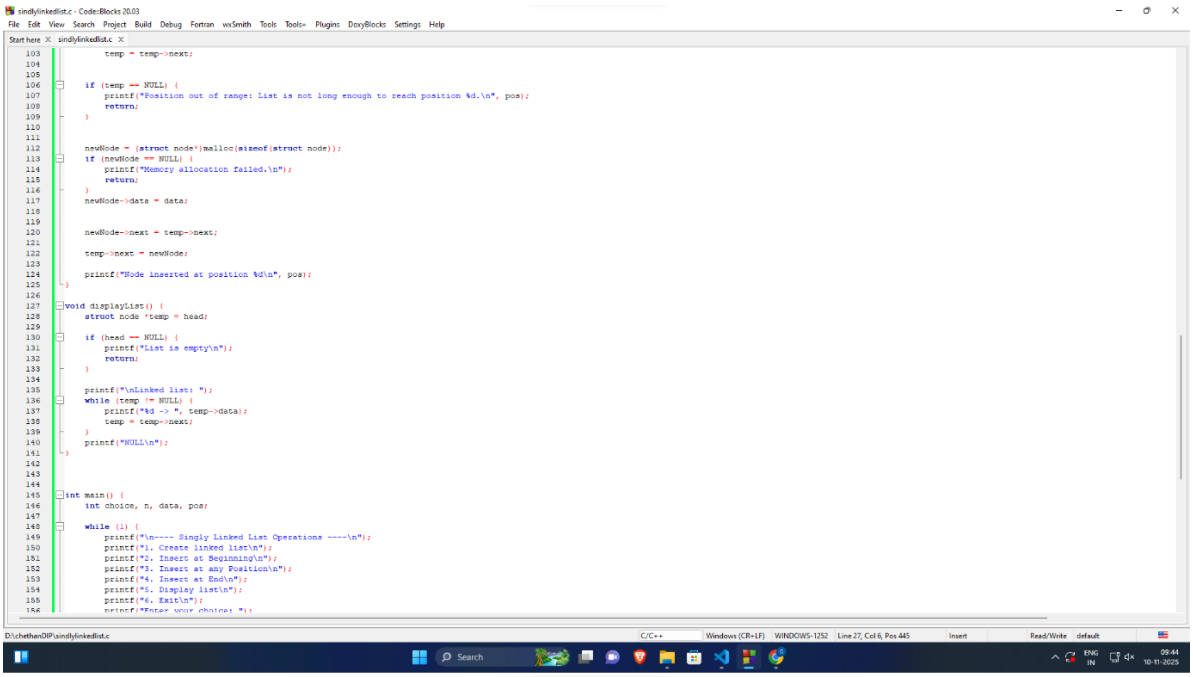


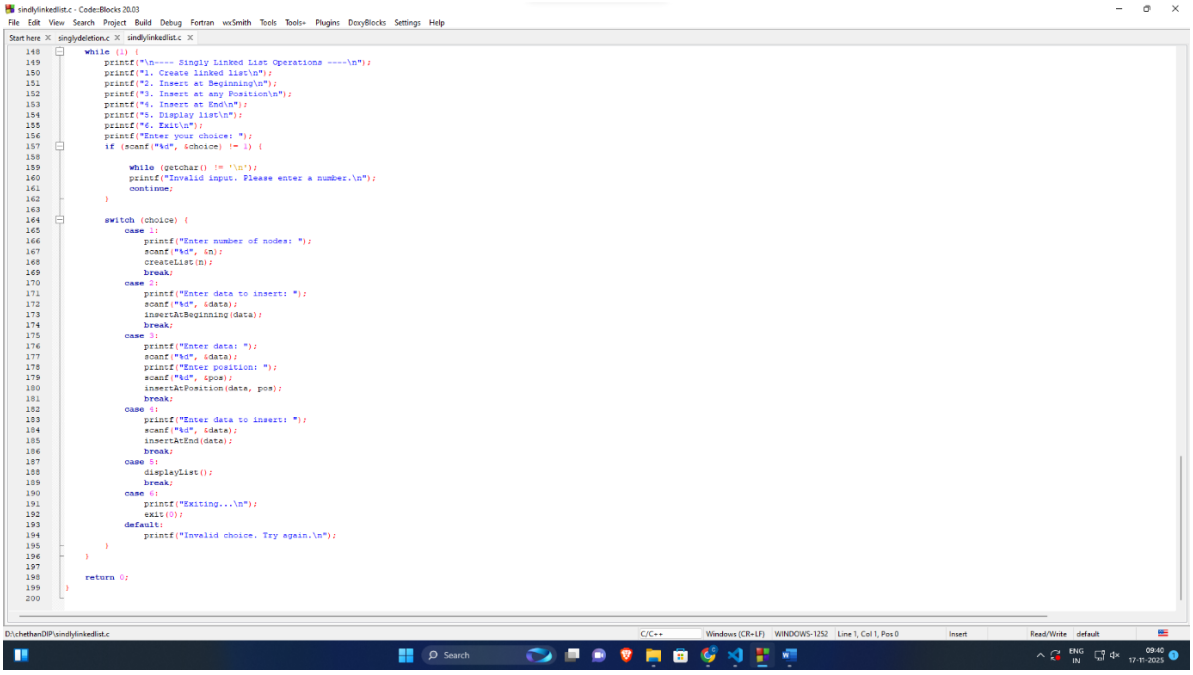
```c
#include <stdio.h>
#include <stdlib.h>

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: Overwriting existing list.\n");
        head = NULL;
    }

    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;
    }

    printf("\nLinked list created successfully\n");
}

void insertAtBeginning(int data) {
    struct node *newNode = (struct node*)malloc(sizeof(struct node));
    if (newNode == NULL) {
```

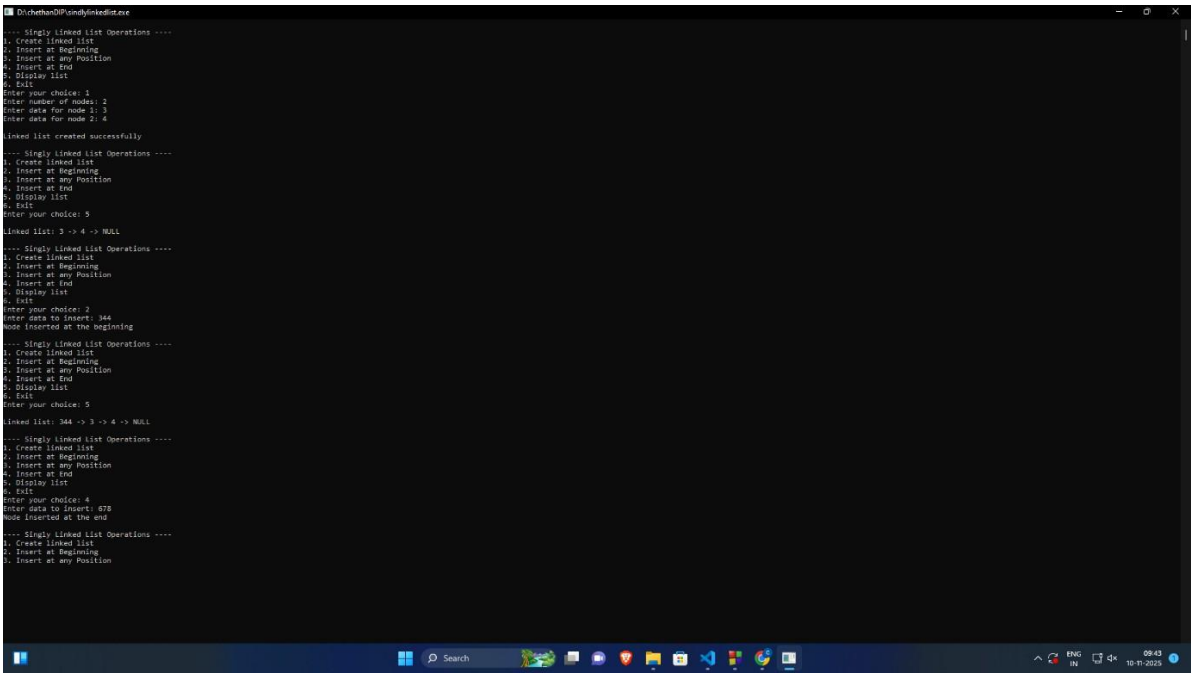
```c
void insertAtBeginning(int data) {
    struct node *newNode = (struct node*)malloc(sizeof(struct node));
    if (newNode == NULL) {
        printf("Memory allocation failed.\n");
        return;
    }
    newNode->data = data;
    newNode->next = head;
    head = newNode;
    printf("Node inserted 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");
}

void insertAtPosition(int data, int pos) {
    int i;
    struct node *newNode, *temp = head;

    if (pos < 1) {
        printf("Invalid position. Position must be 1 or greater.\n");
        return;
    }

    if (pos == 1) {
        insertAtBeginning(data);
        return;
    }


    for (i = 1; i < pos - 1 && temp != NULL; i++)
        temp = temp->next;
```

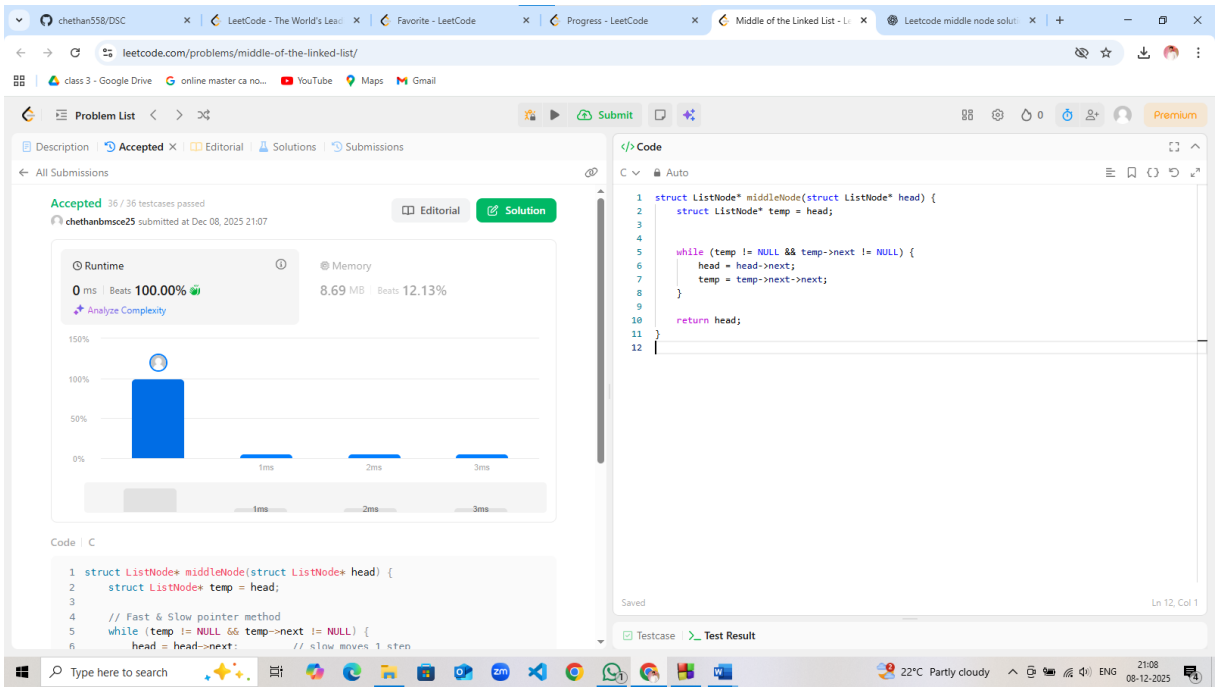


```
        temp = temp->next;

    if (temp == NULL) {
        printf("Position out of range: List is not long enough to reach position %d.\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 %d\n", pos);
}

void displayList() {
    struct node *temp = head;

    if (head == NULL) {
        printf("List is empty\n");
        return;
    }

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



```
    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:\chethanDIP\singlylinkedlist.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


leetcode.com/problems/middle-of-the-linked-list/
Problem List
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 36 / 36 testcases passed
chethanbmsce25 submitted at Dec 08, 2025 21:07
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
8.69 MB | Beats 12.13%
Code
struct ListNode* middleNode(struct ListNode* head) {
struct ListNode* temp = head;
// Fast & Slow pointer method
while (temp != NULL && temp->next != NULL) {
head = head->next; // slow moves 1 step
Code
C
Auto
struct ListNode* middleNode(struct ListNode* head) {
struct ListNode* temp = head;
while (temp != NULL && temp->next != NULL) {
head = head->next;
temp = temp->next->next;
}
return head;
}
Saved
Ln 12, Col 1
Testcase
Test Result
Type here to search
22°C Partly cloudy
ENG
21:08
08-12-2025

DATE: 10/11/25 PAGE:

11) Write a program to implement ~~linked~~ singly linked list with the following a) create a linked list b) Insertion of 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
    create node (value)
    Newnode <- allocate memory for node
    newnode.data <- value
    newnode.next <- null
    return node
end function

=> Insert at first position
    firstposition (value)
    newnode <- create node (value)
    newnode.next <- head
    head <- newnode
end function

=> Insert at end
    insertatend (value)
    newnode -> createnode (value)
    if head = null then
        head <- newnode
        return
    end if
    temp <- head

DATE:

```
    while temp.next ≠ null
        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"
        return
    endif
    newnode ← create node (newvalue)
    newnode.next ← temp.next
    temp.next ← newnode
end
```

=> Display

```
Display ()
temp ← head
while temp ≠ null
    print temp.data
    temp ← temp.next
end while
end function
```

```
#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 node should be greater than 0\n");
        return;
    }
    if ( head != NULL) {
        printf(" Warning: Overwriting existing list.\n");
        head = NULL;
    }
    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;
    }
```

```
    printf("\n Linked list created\n");
}
void insertatBeginning(int data){
    struct node* NewNode = (struct node*) malloc(sizeof(
                                    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(struc
    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");
```

```
void insertAtPosition(int data, int pos) {
    int i;
    struct node * newNode, * temp = head;
    if (pos < 1) {
        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;
    }
```

```
printf("\n Linked list:");
while (temp != NULL) {
    printf("%d ->", temp -> data);
    temp = temp -> next;
}
printf("NULL \n");
}
int main () {
int choice, n, data, pos;
while (1) {
printf("\n 1. creak 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:
        printf("Enter number of nodes:");
        scanf("%d", &n);
        createList(n);
        break;
    case 2:
        printf("Enter data to insert");
        scanf("%d", &data);
        insertAtBeginning(data);
        break;
```

```
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 operations
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:1
Enter number of nodes:3
Enter data for node 1:10
Enter data for node 2:20
```

Enter data for node 3: 30
Enter your choice: 5
10 20 30
Enter your choice: 3
Enter your data: 14
Enter position: 3
Node inserted at position 3
Enter your choice: 4
Enter data to insert: 15
Node inserted at the end

12/11/25