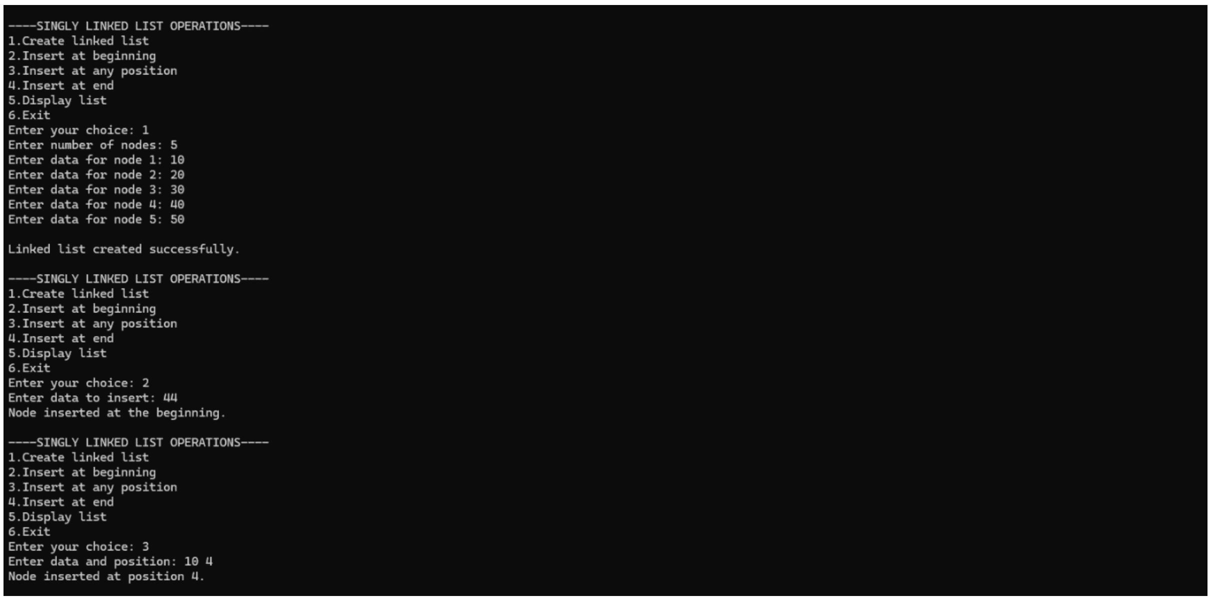
LINKED LIST:

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

// Structure to represent a node
struct Node {
    int data;
    struct Node *next;
};

struct Node *head = NULL;

// Function to create a linked list
void createList(int n) {
    struct Node *newNode, *temp;
    int data, i;

    if (n <= 0) {
        printf("Number of nodes should be greater than 0.\n");
        return;
    }

    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; // first node
        } else {
            temp->next = newNode; // link new node
        }

        temp = newNode; // move temp to last node
    }

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

// Function to insert at beginning
void insertAtBeginning(int data) {
    struct Node *newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = head;
    head = newNode;
    printf("Node inserted at the beginning.\n");
}

// Function to insert at end
void insertAtEnd(int data) {
    struct Node *newNode = (struct Node*)malloc(sizeof(struct Node));
    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");
}

// Function to insert at any position
void insertAtPosition(int data, int pos) {
    int i;
```

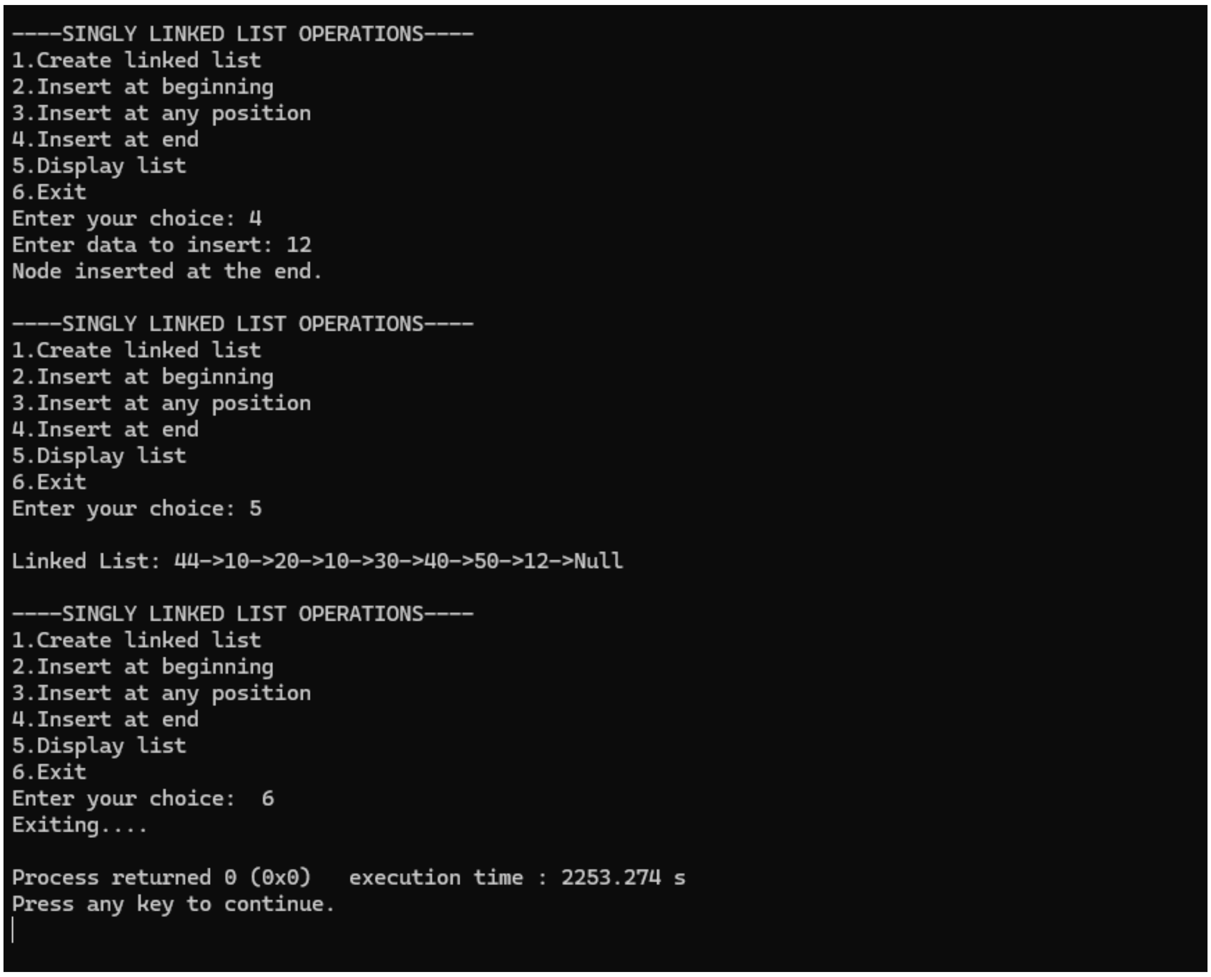
```
struct Node *newNode, *temp = head;

if (pos < 1) {
    printf("Invalid position.\n");
    return;
}

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

newNode = (struct Node*)malloc(sizeof(struct Node));
newNode->data = data;

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

if (temp == NULL) {
    printf("Position out of range.\n");
    free(newNode);
} else {
    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: ");
scanf("%d",&choice);

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 and position: ");
    scanf("%d %d",&data,&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\n");
}
}
return 0;
}
```

OUTPUT:



```
----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: 5
Enter data for node 1: 10
Enter data for node 2: 20
Enter data for node 3: 30
Enter data for node 4: 40
Enter data for node 5: 50

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: 2
Enter data to insert: 44
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: 3
Enter data and position: 10 4
Node inserted at position 4.
```



```
----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: 12
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

Linked List: 44->10->20->10->30->40->50->12->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:  6
Exiting....

Process returned 0 (0x0)   execution time : 2253.274 s
Press any key to continue.
```

OBSERVATION:

10|01|25 PM-4

WAP to implement singly linked list for 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 the linked List.

Pseudocode :-

```
1. Initialize Head = Null
2. Display "Linked List created Successfully".

3. Insert at Beginning :-
   -> Create a new node called new_node
   -> Set New_node.data = value
   -> Set New_node.next = Head
   -> Set head = new_node

4. Insert at any position :-
   -> Input the position & the data.
   -> Create a new node new_node.
   -> Set new_node.data = value
   -> If position = 1,
        Call the "Insert at Beginning".
        & ~~Go to~~ And stop.
   -> Set Temp = head & count = 1.
   -> Repeat while temp ≠ Null & count < position - 1
            Set temp = temp.next
            Increase count = count + 1.
   -> If temp = Null,
        Display "Invalid position"
        ~~Go to~~ And stop.
```

Date __/__/__
Page __

```
-> Set new_node.next = temp.next
-> Set temp.next = New_node

5. Insert at End:
  -> Create a new node
  -> Set new_node.data = value
  -> Set new_node.next = Null
  -> If Head = Null,
         Set head = new_node
         And Stop.
  -> Set temp = head
  -> Repeat while temp.next ≠ Null
         set temp = temp.next
  -> Set temp.next = new_node
  -> Display "Node inserted at End"

6. Display the linked list :
  -> If head = null,
         Display "List is empty".
         And stop.
  -> Set temp = head
  -> Display "Listked List Elements are : "
  -> Repeat while temp ≠ null
         display temp.data
         set temp = temp.next
  -> Display "End of List".
```

[illegible] 10/11/25

```
#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;
        int data, i;

        if (n<=0) {
              printf ("Number of nodes should
                    be greater than 0. \n");
              return;
        }

   for (i=1; i<=n; i++) {
           newNode = (Struct Node *) malloc
                   (size of (Struct Node));
           if (newNode == NULL) {
                printf ("Memory allocation
                        failed. \n");
                return;
           }

      printf ("Enter data for node %d: ", i);
       scanf ("%d", &data);
      newNode->data = data;
```

Date __/__/__
Page ______

```
    newNode->next = NULL;

    if (head == NULL) {
        head = newNode;
    }
    else {
        temp->next = newNode;
    }
    temp = newNode;
  }
  printf("\n Linked list created successfully.\n");
}


void insertAtBeginning (int data) {
    struct Node * newNode = (Struct Node *)
              malloc (sizeof (struct Node));
    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 (size of (struct Node));
    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 insertAtAnyPosition (int data, int pos) {
     int i;
     struct Node * newNode, *temp = head;

     if (pos <1)  {
        printf ("Invaild position. \n");
        return;
     }

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

  newNode = (struct Node*) mallac (size of (
                                   struct Node));
  newNode->data = data;

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

  if (temp == NULL)  {
```

Date __/__/__
Page ______

```
printf ("Position out of range. \n");
    free (newNode);
  }
else {
  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 ("Linked List: ");
  while (temp != NULL)  {
  printf ("%d -> ", temp -> data);
  temp = temp -> ~~dat~~ next;
  }
 printf ("NULL \n");
}

int main () {
 int choice, n, data, pos;

while (1) {
printf ("\n --- Singly Linked List ---- \n");
printf ("1. Create Linked List \n");
printf ("2. Insert at beginning \n");
printf ("3. Insert at any position \n");
```

Date__/__/__
Page_____

```
printf ("4. Insert an end \n");
printf ("5. Display List \n");
printf ("6. Exit \n");
printf ("Enter your choice: ");
scanf ("%d", &choice);

Switch (choice) {.
case 1:
      printf ("Enter no. of nodes: ");
       scanf ("%d", &n);
        createList (n);
         break;

Case 2:
      printf ("Enter data to insert: ");
       scanf (" %d", &data);
      insertAt Beginning (data);
       break;

 case 3:
       printf ("Enter data & position: ");
        scanf ("%d %d", &data, &pos);
       insertAtAnyPosition (data, pos);
        break;

 case 4:
         printf ("Enter data to insert: ");
          Scanf ("%d", &data);
          insertAt End (data);
           break;

case 5:
         displayList ();
          break;
case 6:
    ~~display~~
    printf("Exiting ... \n");
    exit(0);

default:
    printf("Invalid Choice \n");
  }
}
 return 0;
}.
```

Output:

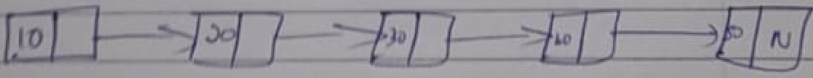
```
---- Singly Linked List operations ----
1. Create Linked List
2. Insert at beginning
3. Insert at any position
4. Insert at end
5. Display list
6. Exit
Enter your choice : 1
Enter no. of nodes : 5
Enter data for node 1: 10
Enter data for node 2: 20
Enter data for node 3: 30
Enter data for node 4: 40
Enter data for node 5: 50

Linked list created succesfully.
```

Date __/__/__
Page ______

```
Enter your choice: 2
Enter data to insert: 44
Node inserted at the beginning.

Enter your choice : 3
Enter data & position : 10 4
Node inserted at position 4.

Enter your choice: 4
Enter data to insert: 12
Node inserted at the end

Enter your choice: 5.
Linked List : 44 -> 10 -> 20 -> 10 -> 30 -> 40 -> 50 ->
                                  12 -> NULL

Enter your choice: 6
Exiting....
```

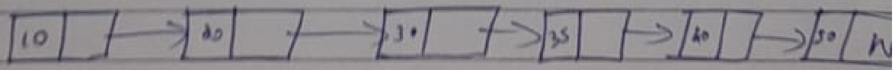
Trace the code

Inserting 10, 20, 30, 40, 50



[illegible] 10/11/25

Insert 35 at 4th position



Insert 55 at the end