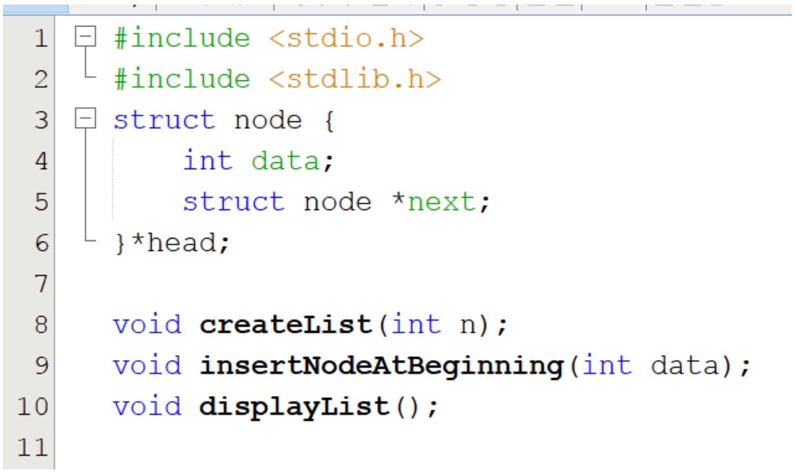
Laboratory 7

1. Questions

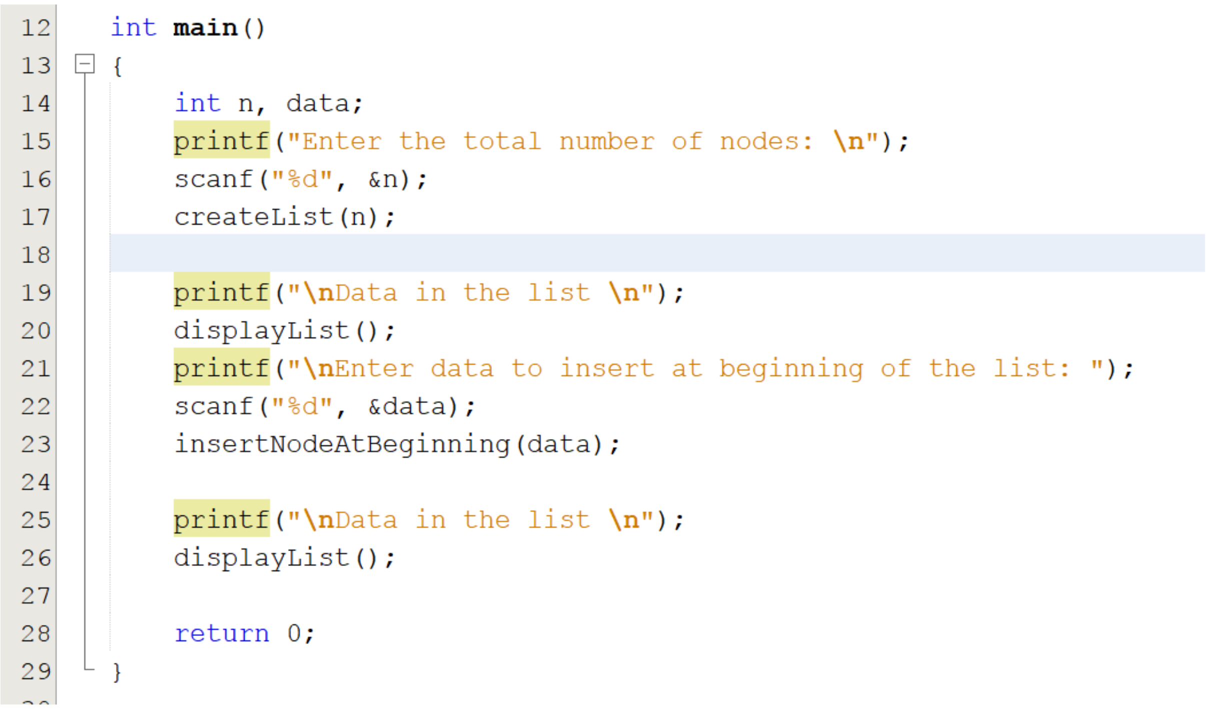
   1. Implement a linked list and illustrate the following operations.
      i. Insert a node at the beginning
      ii. Insert a node at the end
      iii. Print the linked list
   2. Write a program to create a linked list and delete the element entered by a user.

2. Program

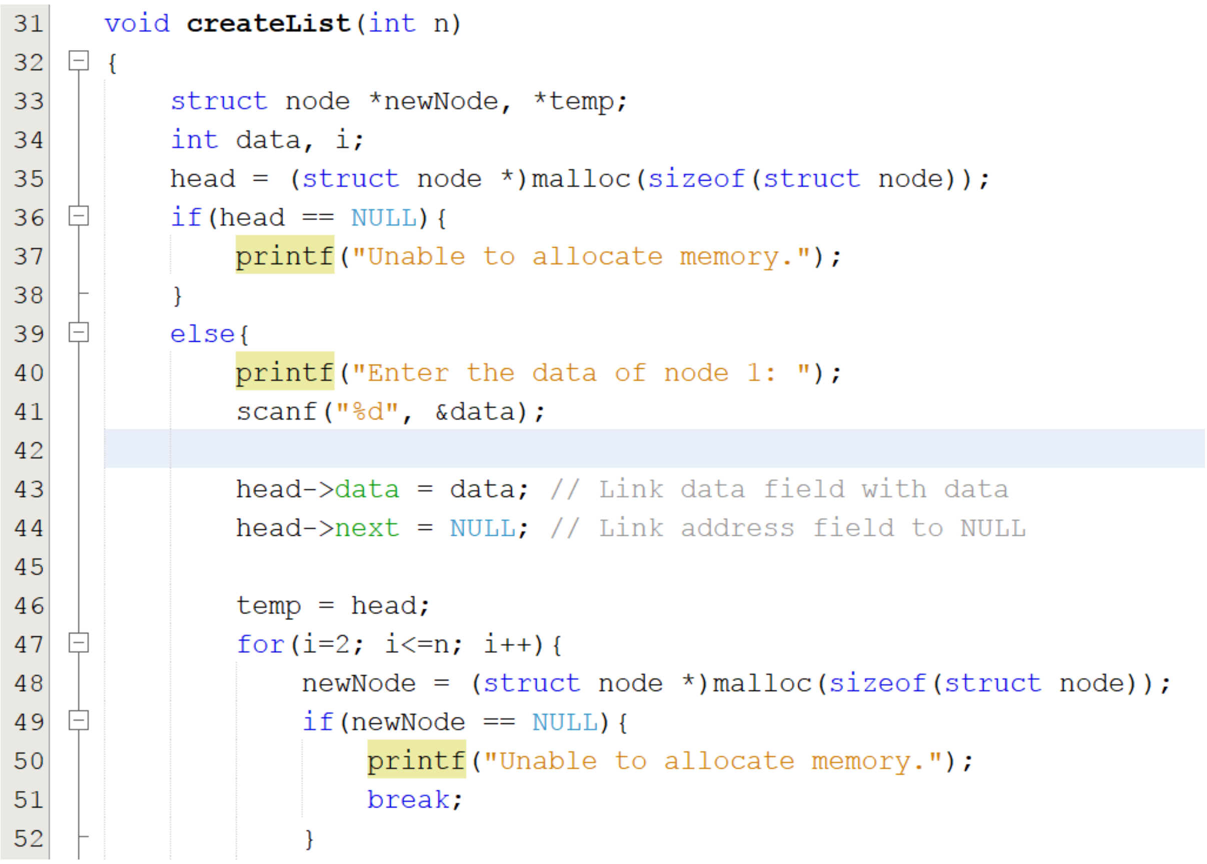
1.1)



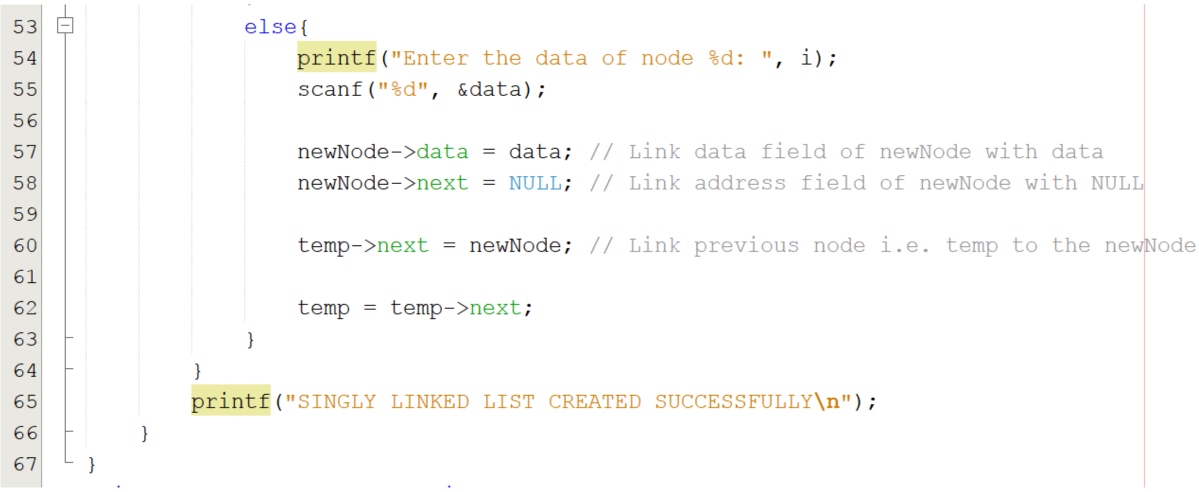
```c
#include <stdio.h>
#include <stdlib.h>
struct node {
    int data;
    struct node *next;
}*head;

void createList(int n);
void insertNodeAtBeginning(int data);
void displayList();

```



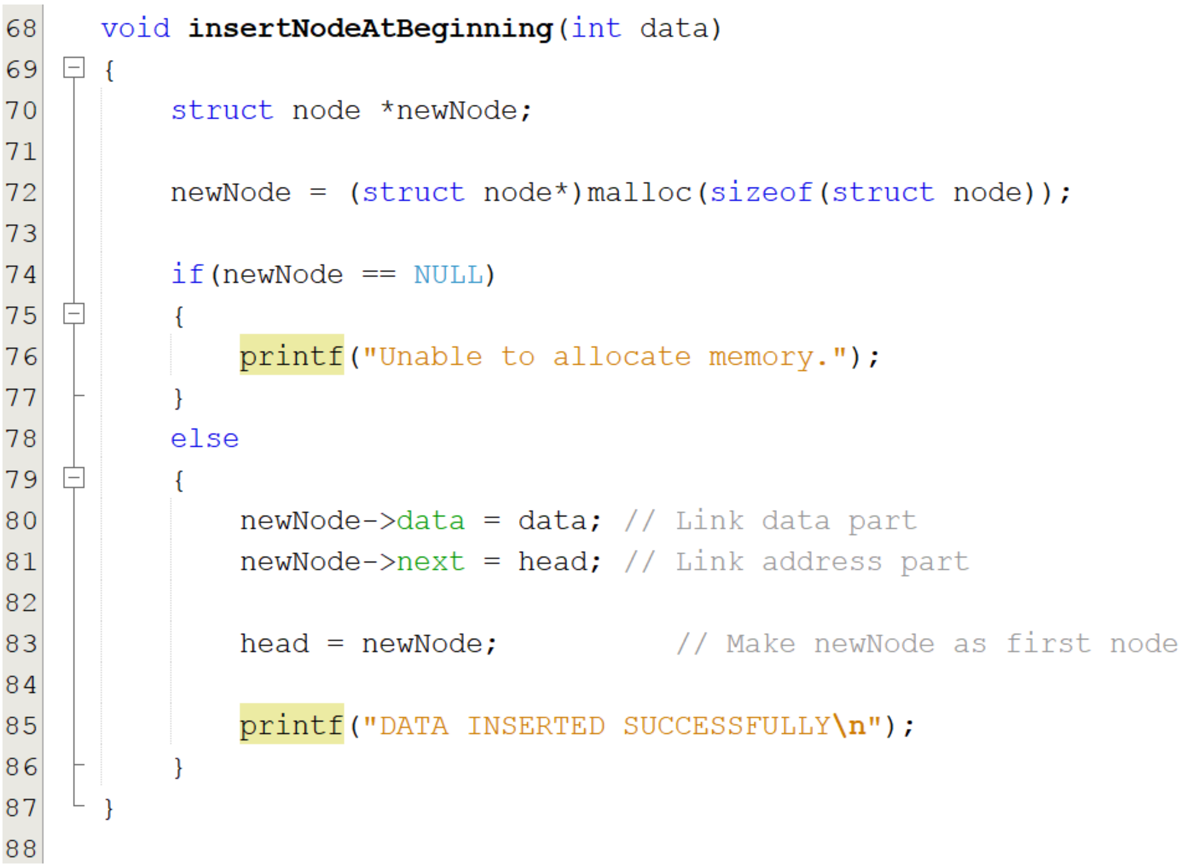
```c
int main()
{
    int n, data;
    printf("Enter the total number of nodes: \n");
    scanf("%d", &n);
    createList(n);

    printf("\nData in the list \n");
    displayList();
    printf("\nEnter data to insert at beginning of the list: ");
    scanf("%d", &data);
    insertNodeAtBeginning(data);

    printf("\nData in the list \n");
    displayList();

    return 0;
}
```



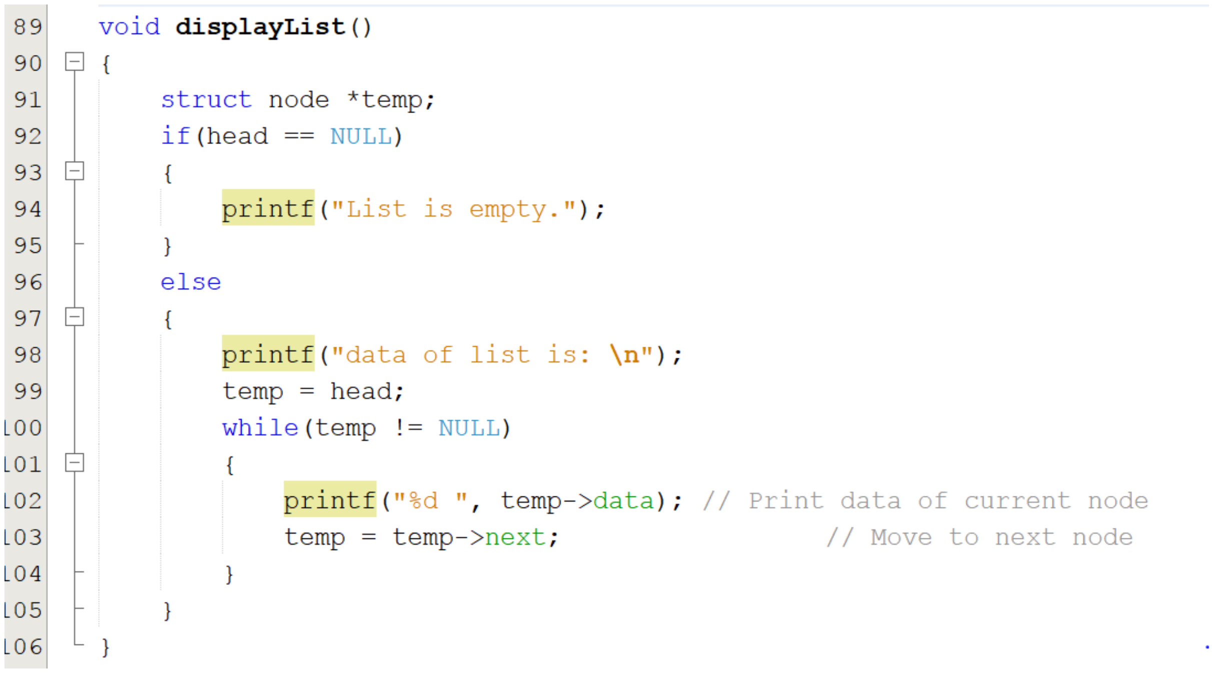
```c
void createList(int n)
{
    struct node *newNode, *temp;
    int data, i;
    head = (struct node *)malloc(sizeof(struct node));
    if(head == NULL){
        printf("Unable to allocate memory.");
    }
    else{
        printf("Enter the data of node 1: ");
        scanf("%d", &data);

        head->data = data; // Link data field with data
        head->next = NULL; // Link address field to NULL

        temp = head;
        for(i=2; i<=n; i++){
            newNode = (struct node *)malloc(sizeof(struct node));
            if(newNode == NULL){
                printf("Unable to allocate memory.");
                break;
            }
```



```c
            else{
                printf("Enter the data of node %d: ", i);
                scanf("%d", &data);

                newNode->data = data; // Link data field of newNode with data
                newNode->next = NULL; // Link address field of newNode with NULL

                temp->next = newNode; // Link previous node i.e. temp to the newNode

                temp = temp->next;
            }
        }
        printf("SINGLY LINKED LIST CREATED SUCCESSFULLY\n");
    }
}
```

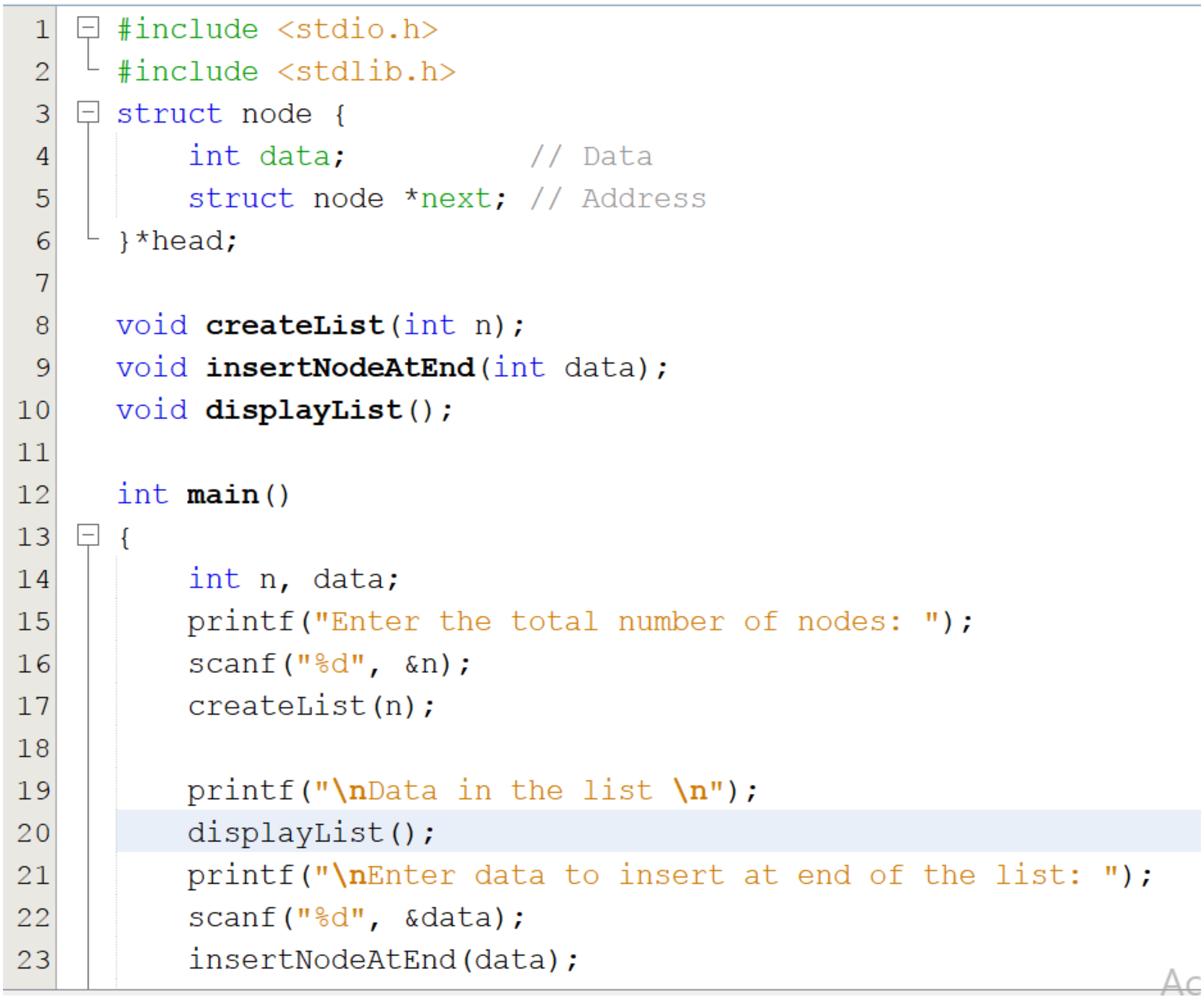


```
void insertNodeAtBeginning(int data)
{
    struct node *newNode;

    newNode = (struct node*)malloc(sizeof(struct node));

    if(newNode == NULL)
    {
        printf("Unable to allocate memory.");
    }
    else
    {
        newNode->data = data; // Link data part
        newNode->next = head; // Link address part

        head = newNode;          // Make newNode as first node

        printf("DATA INSERTED SUCCESSFULLY\n");
    }
}

```

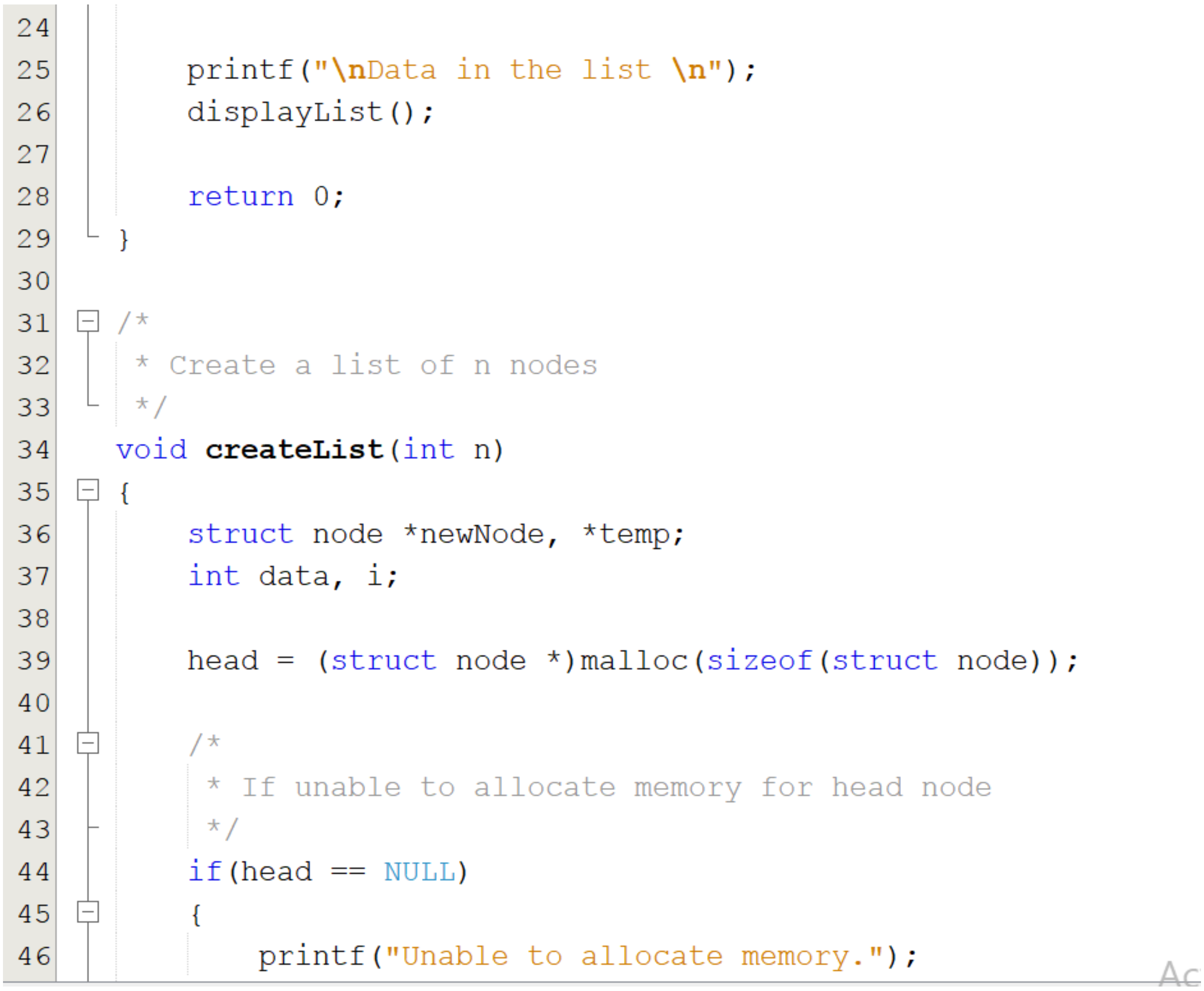


```
void displayList()
{
    struct node *temp;
    if(head == NULL)
    {
        printf("List is empty.");
    }
    else
    {
        printf("data of list is: \n");
        temp = head;
        while(temp != NULL)
        {
            printf("%d ", temp->data); // Print data of current node
            temp = temp->next;                 // Move to next node
        }
    }
}
```

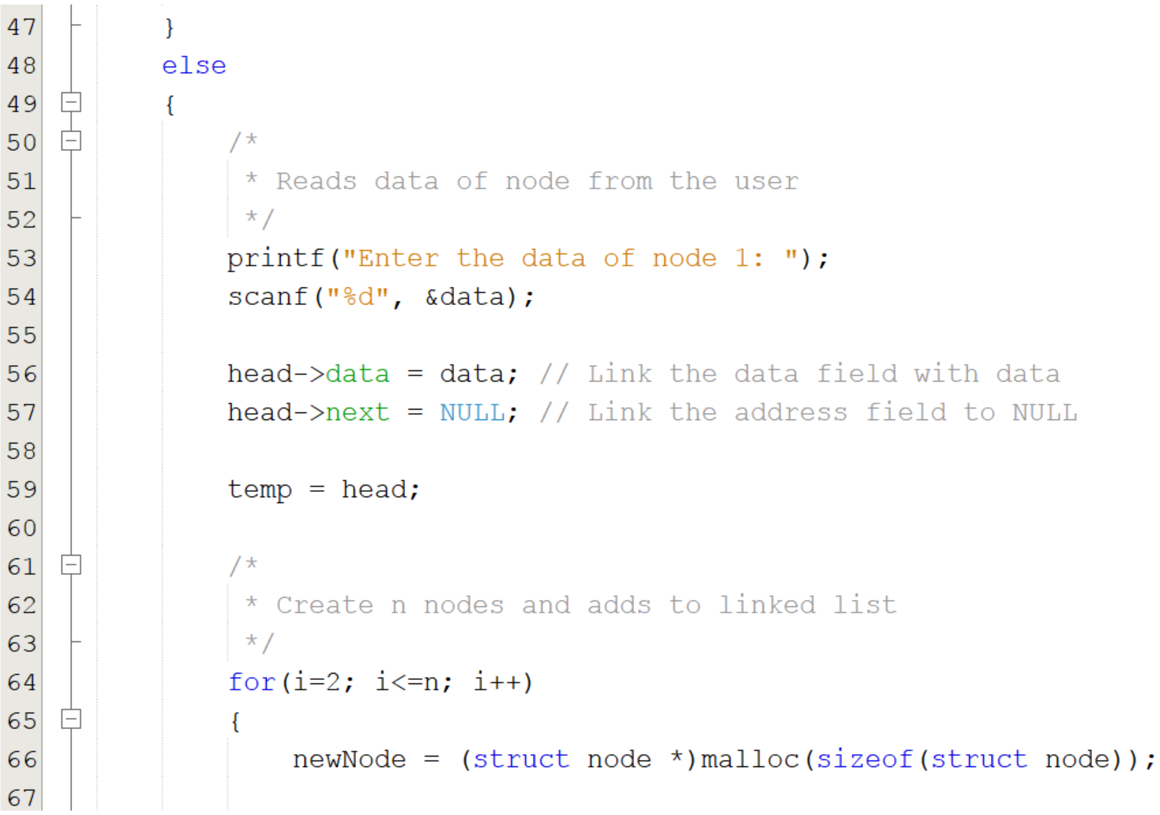
1.2)



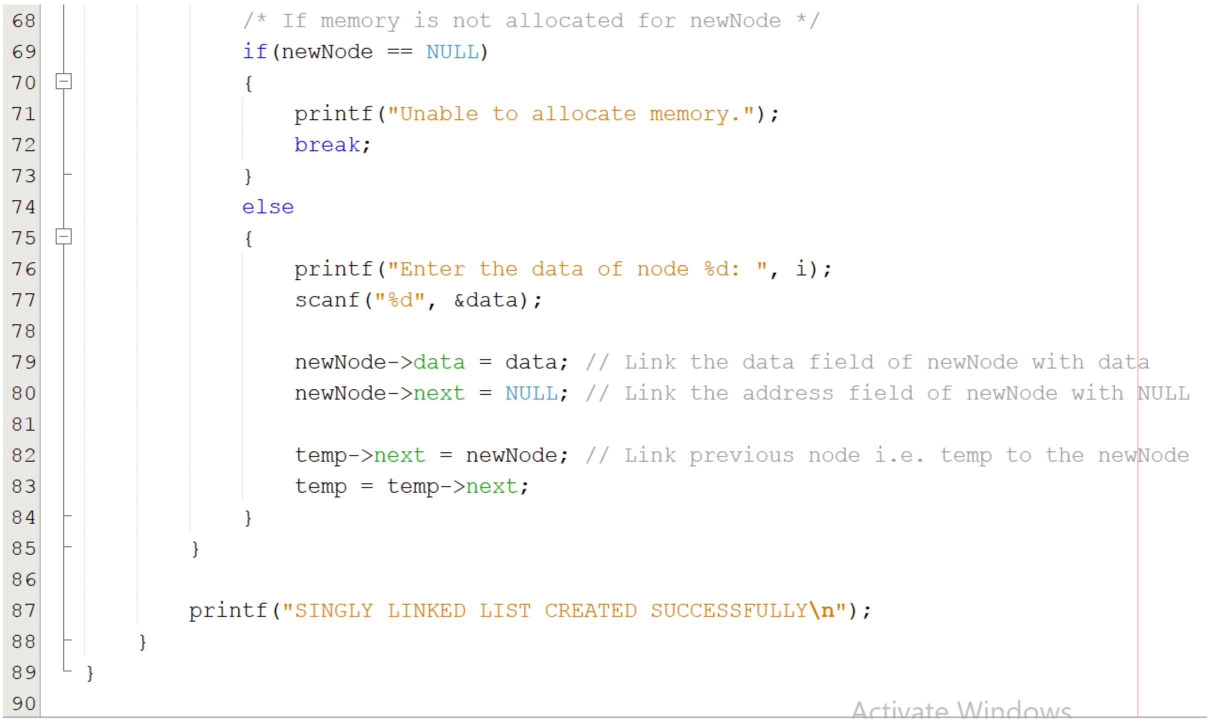
```c
#include <stdio.h>
#include <stdlib.h>
struct node {
    int data;          // Data
    struct node *next; // Address
}*head;

void createList(int n);
void insertNodeAtEnd(int data);
void displayList();

int main()
{
    int n, data;
    printf("Enter the total number of nodes: ");
    scanf("%d", &n);
    createList(n);

    printf("\nData in the list \n");
    displayList();
    printf("\nEnter data to insert at end of the list: ");
    scanf("%d", &data);
    insertNodeAtEnd(data);
```



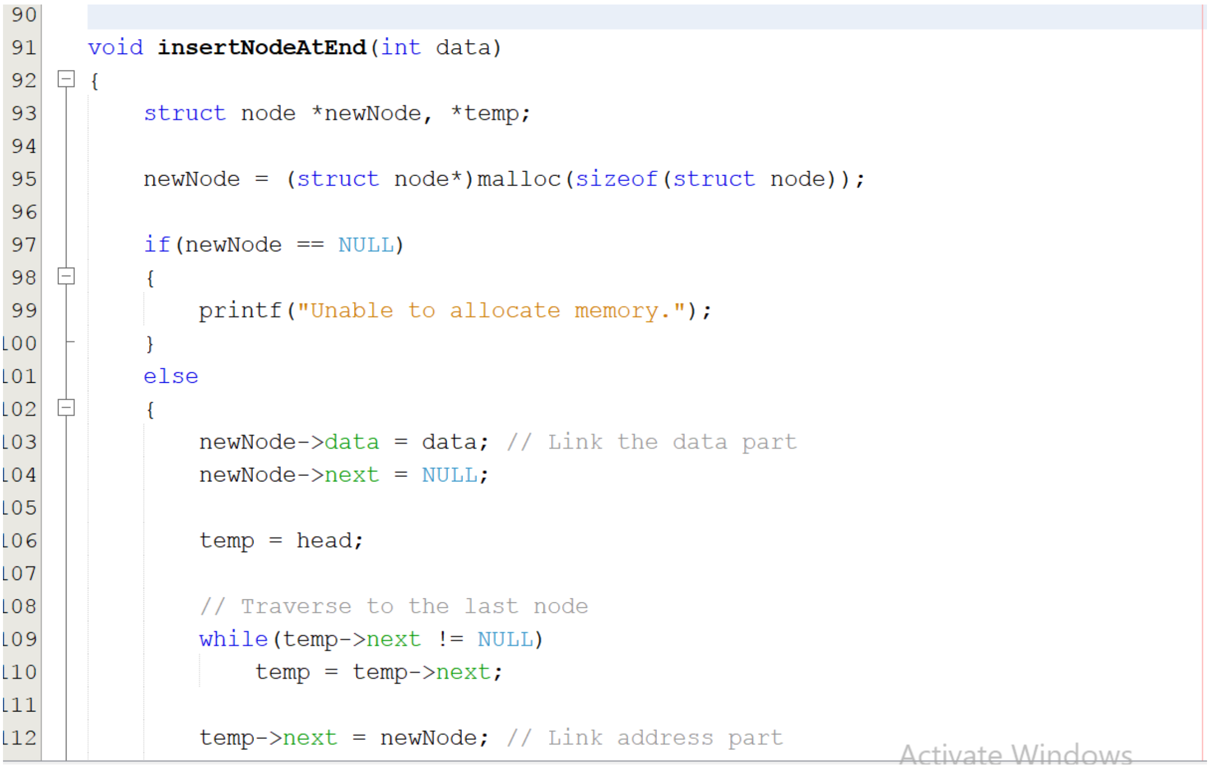
```c

    printf("\nData in the list \n");
    displayList();

    return 0;
}

/*
 * Create a list of n nodes
 */
void createList(int n)
{
    struct node *newNode, *temp;
    int data, i;

    head = (struct node *)malloc(sizeof(struct node));

    /*
     * If unable to allocate memory for head node
     */
    if(head == NULL)
    {
        printf("Unable to allocate memory.");
```



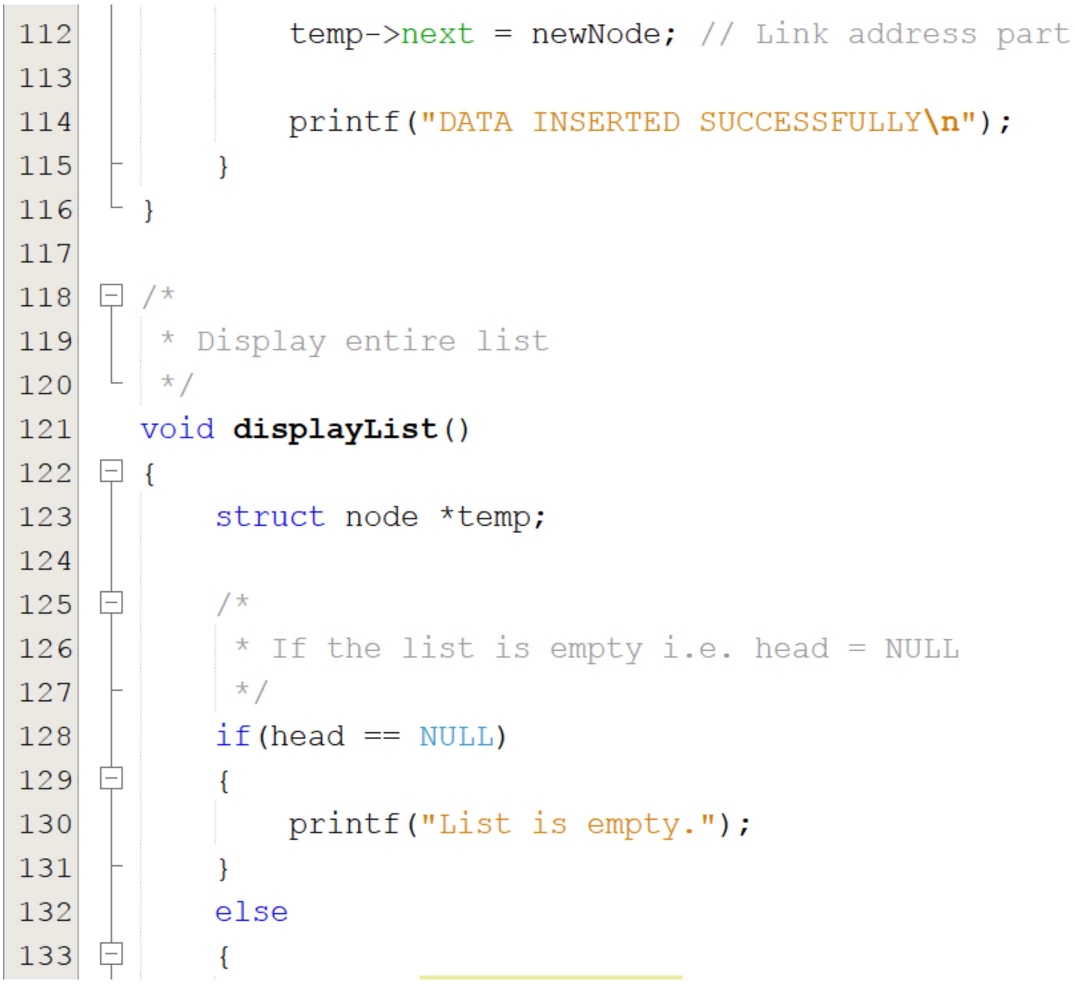
```c
    }
    else
    {
        /*
         * Reads data of node from the user
         */
        printf("Enter the data of node 1: ");
        scanf("%d", &data);

        head->data = data; // Link the data field with data
        head->next = NULL; // Link the address field to NULL

        temp = head;

        /*
         * Create n nodes and adds to linked list
         */
        for(i=2; i<=n; i++)
        {
            newNode = (struct node *)malloc(sizeof(struct node));

```



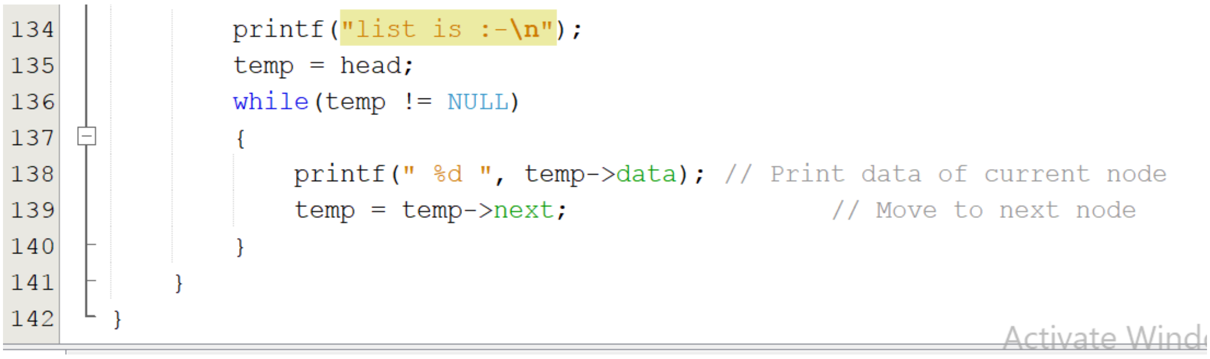
```c
            /* If memory is not allocated for newNode */
            if(newNode == NULL)
            {
                printf("Unable to allocate memory.");
                break;
            }
            else
            {
                printf("Enter the data of node %d: ", i);
                scanf("%d", &data);

                newNode->data = data; // Link the data field of newNode with data
                newNode->next = NULL; // Link the address field of newNode with NULL

                temp->next = newNode; // Link previous node i.e. temp to the newNode
                temp = temp->next;
            }
        }

        printf("SINGLY LINKED LIST CREATED SUCCESSFULLY\n");
    }
}
```



```c

void insertNodeAtEnd(int data)
{
    struct node *newNode, *temp;

    newNode = (struct node*)malloc(sizeof(struct node));

    if(newNode == NULL)
    {
        printf("Unable to allocate memory.");
    }
    else
    {
        newNode->data = data; // Link the data part
        newNode->next = NULL;

        temp = head;

        // Traverse to the last node
        while(temp->next != NULL)
            temp = temp->next;

        temp->next = newNode; // Link address part
```

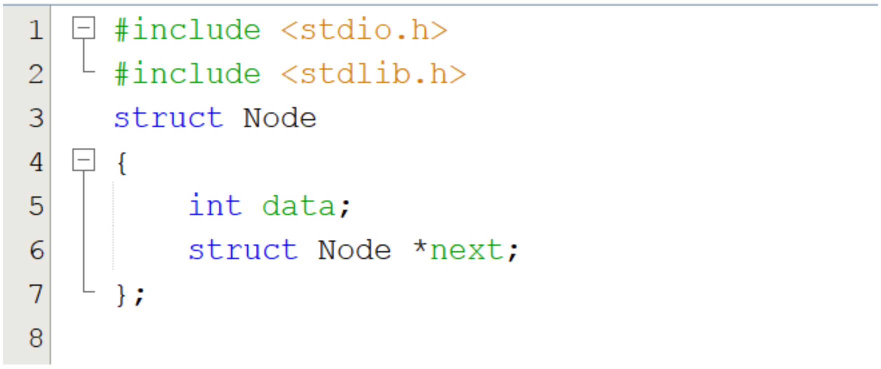


```
        temp->next = newNode; // Link address part

        printf("DATA INSERTED SUCCESSFULLY\n");
    }
}

/*
 * Display entire list
 */
void displayList()
{
    struct node *temp;

    /*
     * If the list is empty i.e. head = NULL
     */
    if(head == NULL)
    {
        printf("List is empty.");
    }
    else
    {
```

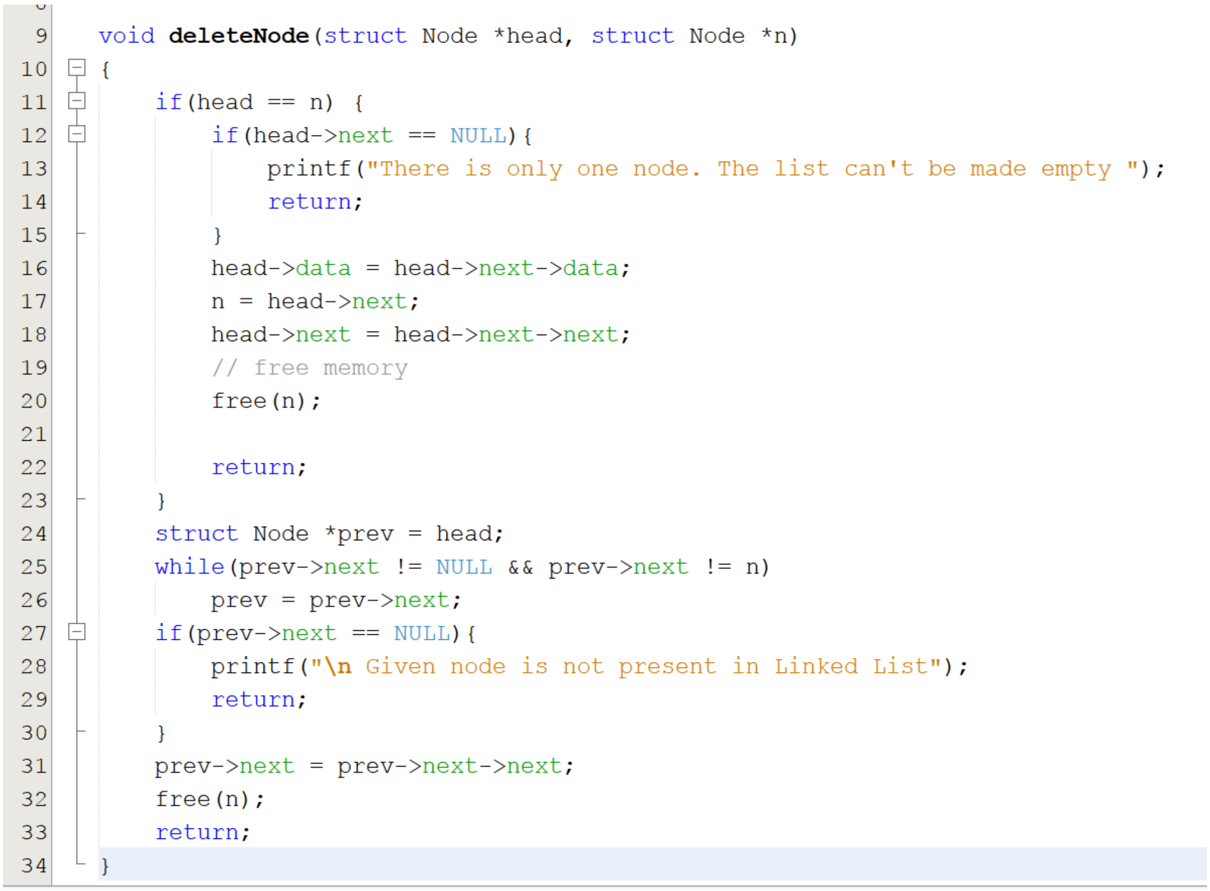


```
        printf("list is :-\n");
        temp = head;
        while(temp != NULL)
        {
            printf(" %d ", temp->data); // Print data of current node
            temp = temp->next;                 // Move to next node
        }
    }
}
```

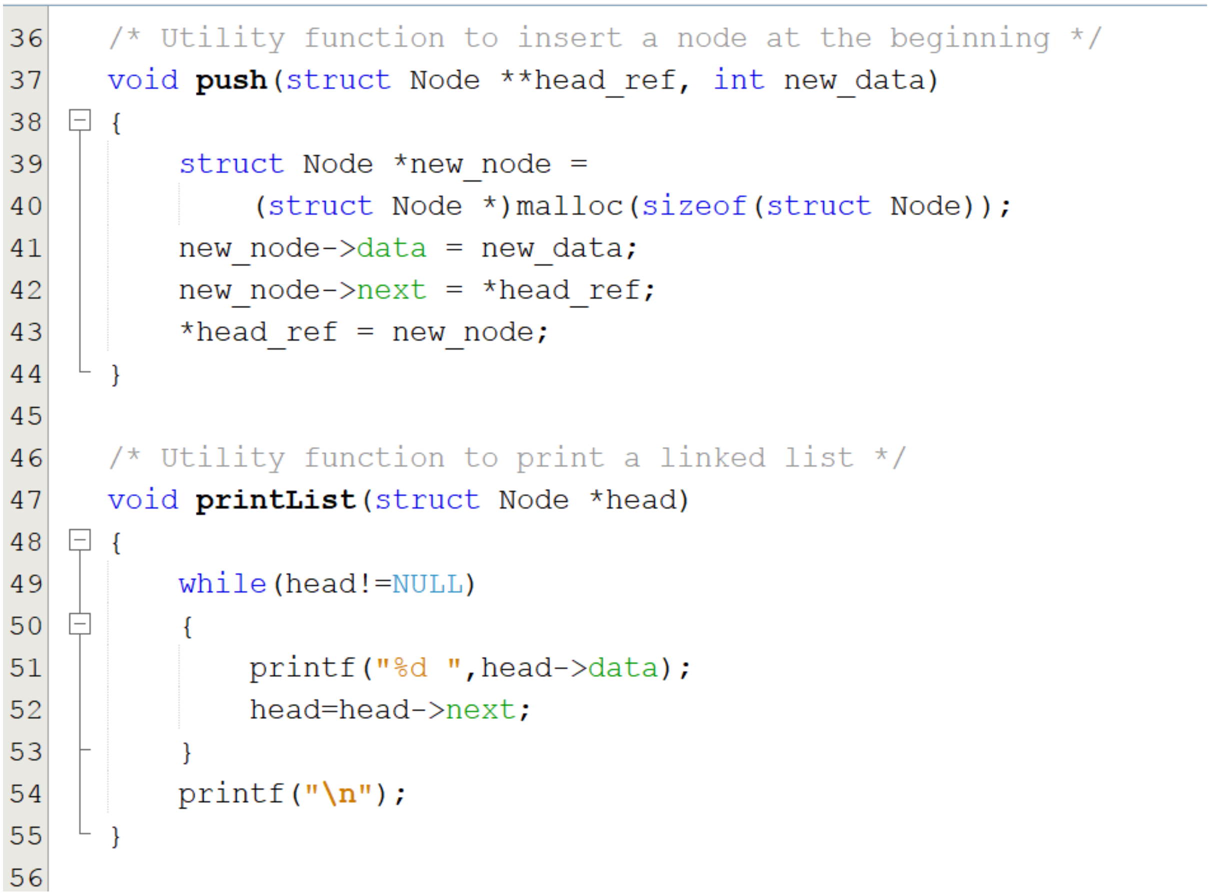
1.3)



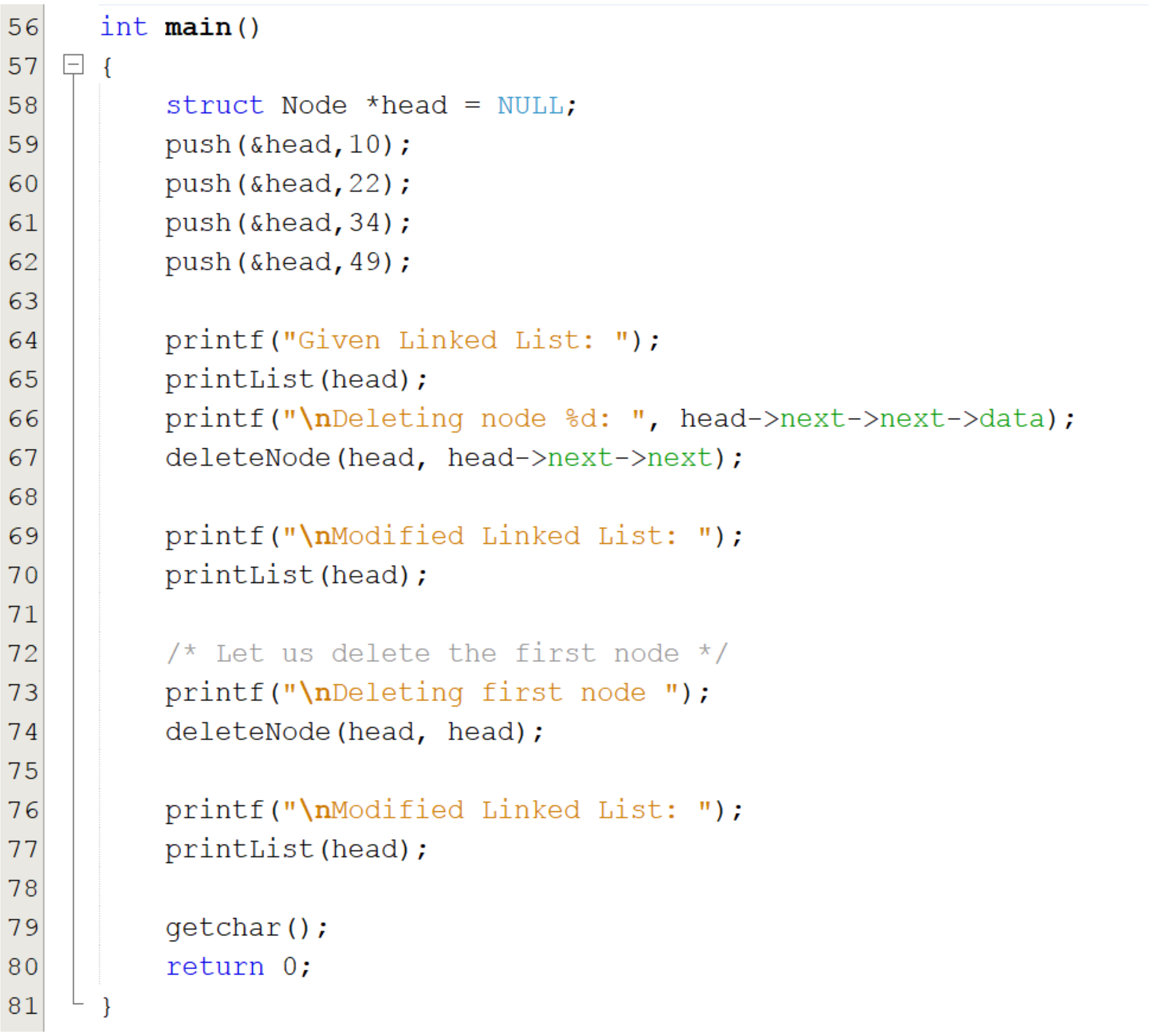
```c
#include <stdio.h>
#include <stdlib.h>
struct Node
{
    int data;
    struct Node *next;
};

```



```c
void deleteNode(struct Node *head, struct Node *n)
{
    if(head == n)  {
        if(head->next == NULL){
            printf("There is only one node. The list can't be made empty ");
            return;
        }
        head->data = head->next->data;
        n = head->next;
        head->next = head->next->next;
        // free memory
        free(n);

        return;
    }
    struct Node *prev = head;
    while(prev->next != NULL && prev->next != n)
        prev = prev->next;
    if(prev->next == NULL){
        printf("\n Given node is not present in Linked List");
        return;
    }
    prev->next = prev->next->next;
    free(n);
    return;
}
```



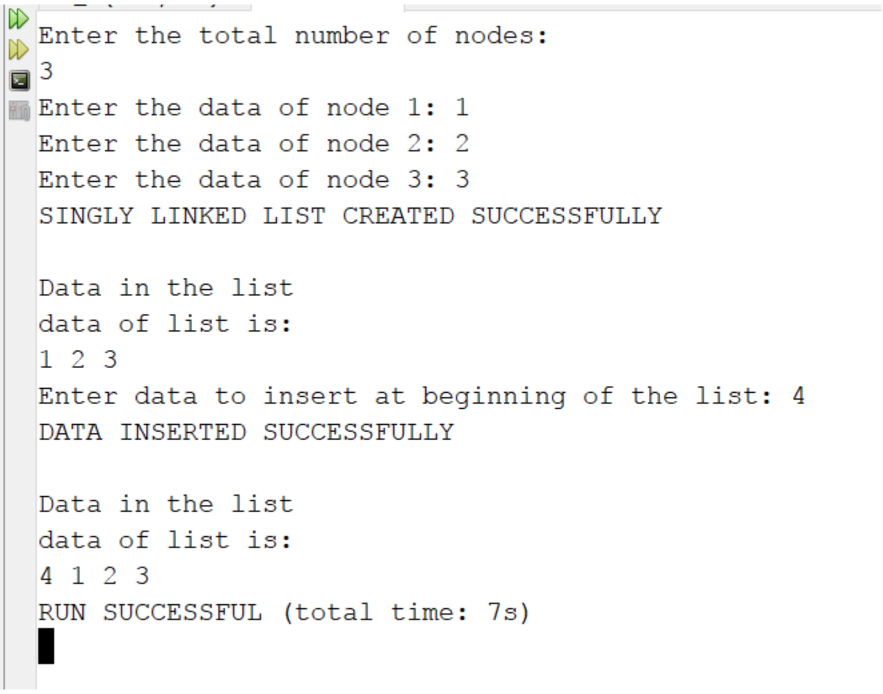
```c
/* Utility function to insert a node at the beginning */
void push(struct Node **head_ref, int new_data)
{
    struct Node *new_node =
        (struct Node *)malloc(sizeof(struct Node));
    new_node->data = new_data;
    new_node->next = *head_ref;
    *head_ref = new_node;
}

/* Utility function to print a linked list */
void printList(struct Node *head)
{
    while(head!=NULL)
    {
        printf("%d ",head->data);
        head=head->next;
    }
    printf("\n");
}

```



```c
int main()
{
    struct Node *head = NULL;
    push(&head,10);
    push(&head,22);
    push(&head,34);
    push(&head,49);

    printf("Given Linked List: ");
    printList(head);
    printf("\nDeleting node %d: ", head->next->next->data);
    deleteNode(head, head->next->next);

    printf("\nModified Linked List: ");
    printList(head);

    /* Let us delete the first node */
    printf("\nDeleting first node ");
    deleteNode(head, head);

    printf("\nModified Linked List: ");
    printList(head);

    getchar();
    return 0;
}
```

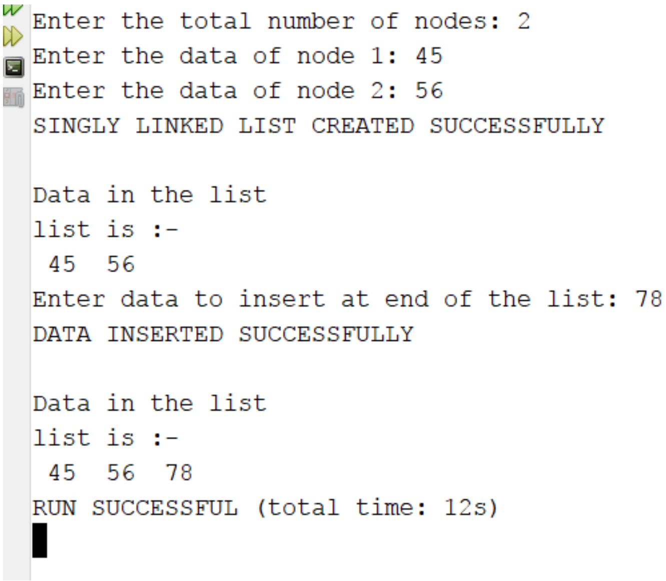
3. Presentation of Results

1.1) and 1.3)



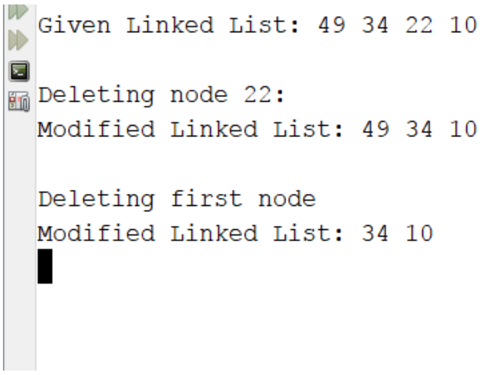
```
Enter the total number of nodes:
3
Enter the data of node 1: 1
Enter the data of node 2: 2
Enter the data of node 3: 3
SINGLY LINKED LIST CREATED SUCCESSFULLY

Data in the list
data of list is:
1 2 3
Enter data to insert at beginning of the list: 4
DATA INSERTED SUCCESSFULLY

Data in the list
data of list is:
4 1 2 3
RUN SUCCESSFUL (total time: 7s)
```

1.2)



```
Enter the total number of nodes: 2
Enter the data of node 1: 45
Enter the data of node 2: 56
SINGLY LINKED LIST CREATED SUCCESSFULLY

Data in the list
list is :-
 45  56
Enter data to insert at end of the list: 78
DATA INSERTED SUCCESSFULLY

Data in the list
list is :-
 45  56  78
RUN SUCCESSFUL (total time: 12s)
```

Q2



```
Given Linked List: 49 34 22 10

Deleting node 22:
Modified Linked List: 49 34 10

Deleting first node
Modified Linked List: 34 10
```

4. Conclusions :

   All the programs have been executed successfully.

   Linked lists concepts have been revised.