**OBJECTIVES:**

In this lab we learnt about;

* Linked List
* Types of Linked List

**7.1.1 Linked List**

A linked list is similar in many ways to arrays. It is a series of connected "nodes" that contains the "address" of the next node. Each node can store a data point, which may be a number, a string or any other type of data.

**7.1.2 Linked List Representations**



You have to start somewhere, so we give the address of the first node a special name called HEAD.  
Also, the last node in the linked list can be identified because its next portion points to NULL.  
How another node is referenced?

Some pointer magic is involved. Let's think about what each node contains:

* A data item
* An address of another node

We wrap both the data item and the next node reference in a struct as:

struct node

{

int data;

struct node \*next;

};

Understanding the structure of a linked list node is the key to having a grasp on it.  
Each struct node has a data item and a pointer to another struct node.

**7.1.3 Operations on the Linked List**

Let us create a simple Linked List with three items to understand how this works.

/\* Initialize nodes \*/

struct node \*head;

struct node \*one = NULL;

struct node \*two = NULL;

struct node \*three = NULL;

/\* Allocate memory \*/

one = malloc(sizeof(struct node));

two = malloc(sizeof(struct node));

three = malloc(sizeof(struct node));

/\* Assign data values \*/

one->data = 1;

two->data = 2;

three->data=3;

/\* Connect nodes \*/

one->next = two;

two->next = three;

three->next = NULL;

/\* Save address of first node in head \*/

head = one;

If you didn't understand any of the lines above, all you need is a refresher on pointers and structs.

In just a few steps, we have created a simple linkedlist with three nodes.



The power of linkedlist comes from the ability to break the chain and rejoin it. E.g. if you wanted to put an element 4 between 1 and 2, the steps would be:

* Create a new struct node and allocate memory to it.
* Add its data value as 4
* Point its next pointer to the struct node containing 2 as data value
* Change next pointer of "1" to the node we just created.

Doing something similar in an array would have required shifting the positions of all the subsequent elements.

**7.1.4 Types of Linked List**

There are three common types of Linked List.

* Singly Linked List
* Doubly Linked List
* Circular Linked List

## 

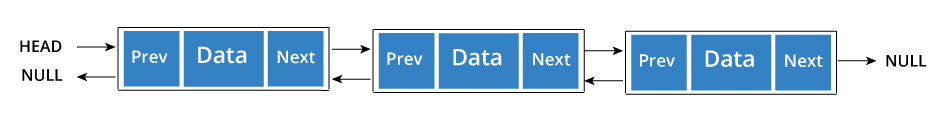
## **7.1.4.1 Singly Linked List**

## It is the most common. Each node has data and a pointer to the next node.



## **7.1.4.2 Doubly Linked List**

## We add a pointer to the previous node in a doubly linked list. Thus, we can go in either direction: forward or backward.



A node is represented as

struct node {

int data;

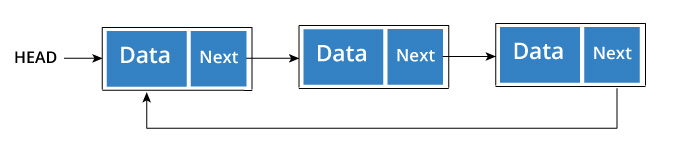
struct node \*next;

struct node \*prev;

}

## **7.1.4.3 Circular Linked List**

## A circular linked list is a variation of linked list in which the last element is linked to the first element. This forms a circular loop.



A circular linked list can be either singly linked or doubly linked.

* for singly linked list, next pointer of last item points to the first item
* In doubly linked list, prev pointer of first item points to last item as well.

**7.1.5 Operations on linked list**

* **Traverse a linked list**

Displaying the contents of a linked list is very simple. We keep moving the temp node to the next one and display its contents.

When temp is NULL, we know that we have reached the end of linked list so we get out of the while loop.

The output of this program will be:

List elements are - 1 --->2 --->3 --->

## **Add elements to linked list**

## You can add elements to beginning, middle or end of linked list.

### Add to beginning

* Allocate memory for new node
* Store data
* Change next of new node to point to head
* Change head to point to recently created node

### Add to end

* Allocate memory for new node
* Store data
* Traverse to last node
* Change next of last node to recently created node

### Add to middle

* Allocate memory and store data for new node
* Traverse to node just before the required position of new node
* Change next pointers to include new node in between

## **Delete from a linked list**

You can delete either from beginning, end or from a particular position.

### Delete from beginning

* Point head to the second node

### Delete from end

* Traverse to second last element
* Change its next pointer to null

### Delete from middle

* Traverse to element before the element to be deleted
* Change next pointers to exclude the node from the chain

**LAB TASKS:**

**TASK 1:**

Implement singly linked list with following operations:

a) Traverse

b) Delete

c) Print

d) Add

**CODE:**



**A screenshot of a computer code

Description automatically generated with low confidence**

**A screenshot of a computer code

Description automatically generated with medium confidence**

**A screen shot of a computer code

Description automatically generated with low confidence**

**A screen shot of a computer program

Description automatically generated with low confidence**

**OUTPUT:**

A screenshot of a computer

Description automatically generated with medium confidence

**TASK 2:**

Implement a single linked list with following operations

a) Insert at start of the list (preappend data)

b) Insertion at end (append data)

c) Insertion at nth location

d) Deletion from start of the list

e) Deletion the end of the list

f) Insertion from nth location

g) Search for a key

h) Update list

i) Empty list

j) Calculate size of the list

**CODE:**

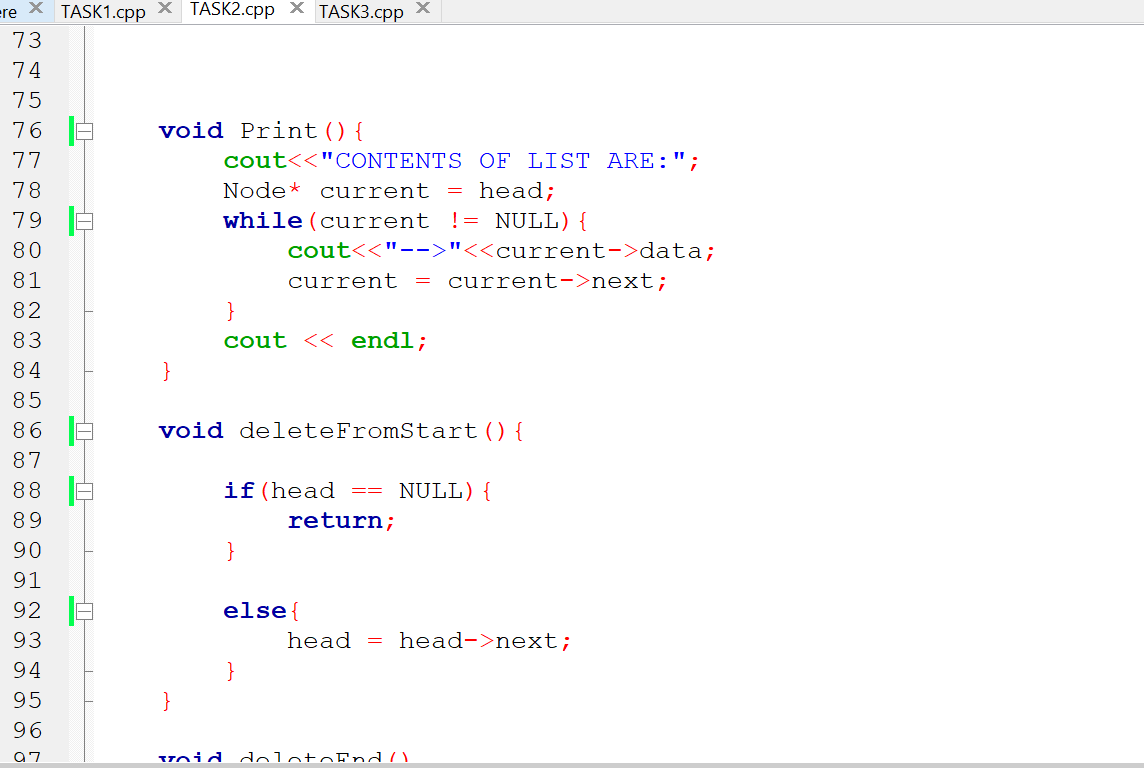
****

**A screenshot of a computer code

Description automatically generated with medium confidence**

**A screenshot of a computer code

Description automatically generated with medium confidence**

****

**A screen shot of a computer code

Description automatically generated with low confidence**

**A screenshot of a computer program

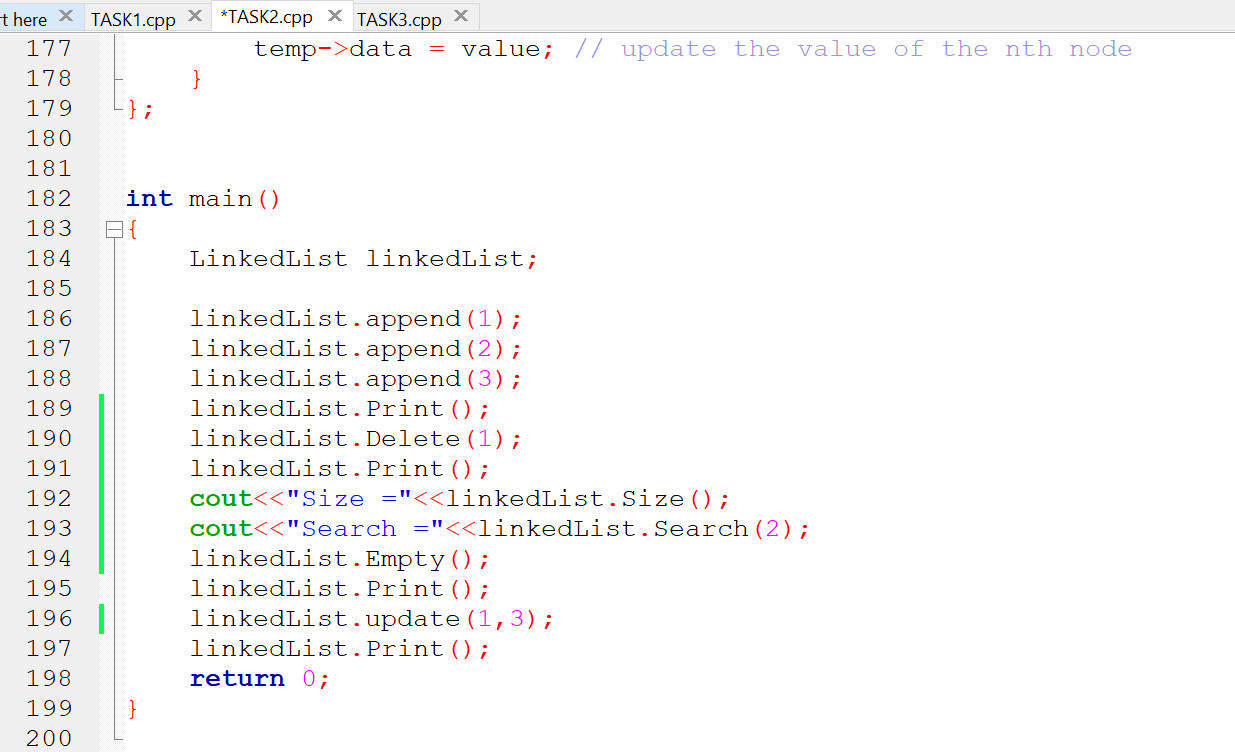
Description automatically generated with medium confidence**

**A screenshot of a computer program

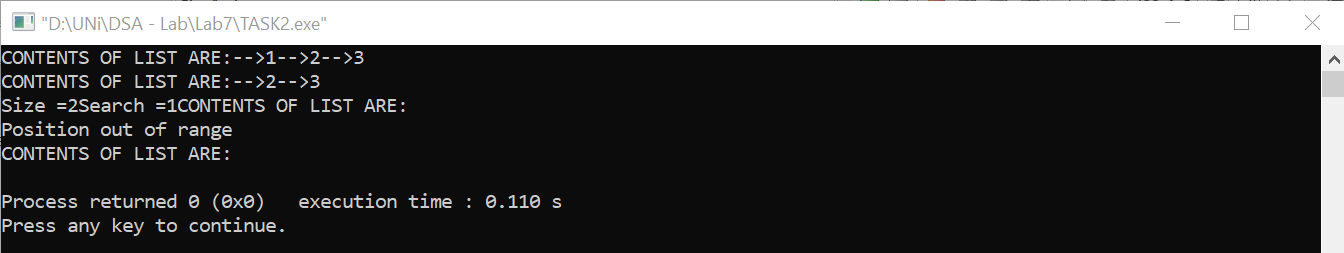
Description automatically generated with medium confidence**

**A screenshot of a computer screen

Description automatically generated with medium confidence**

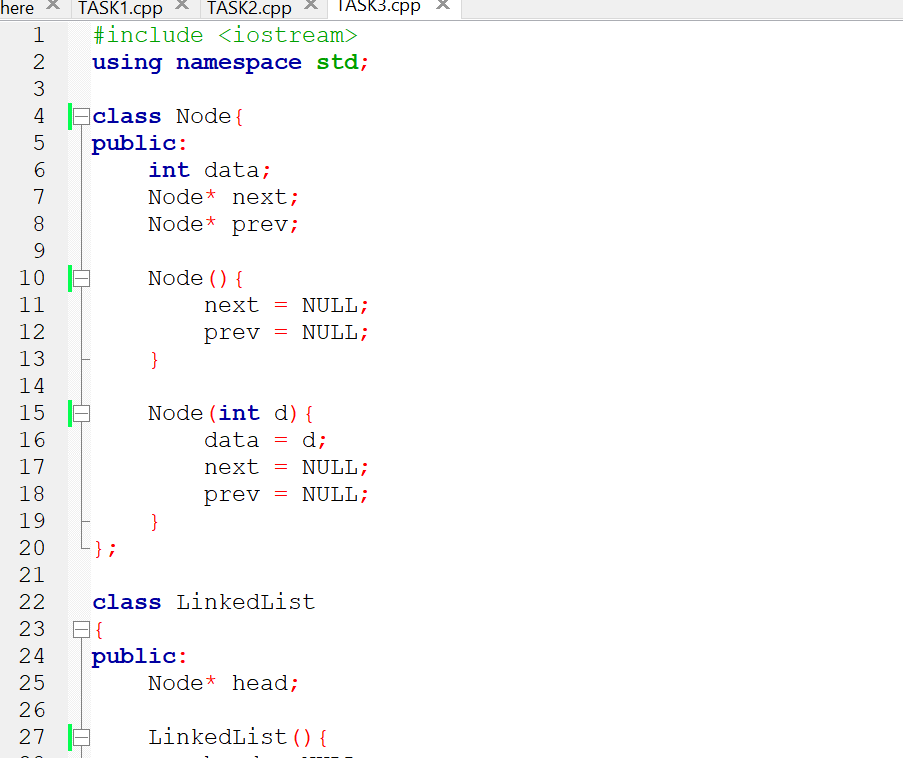
****

**OUTPUT:**

****

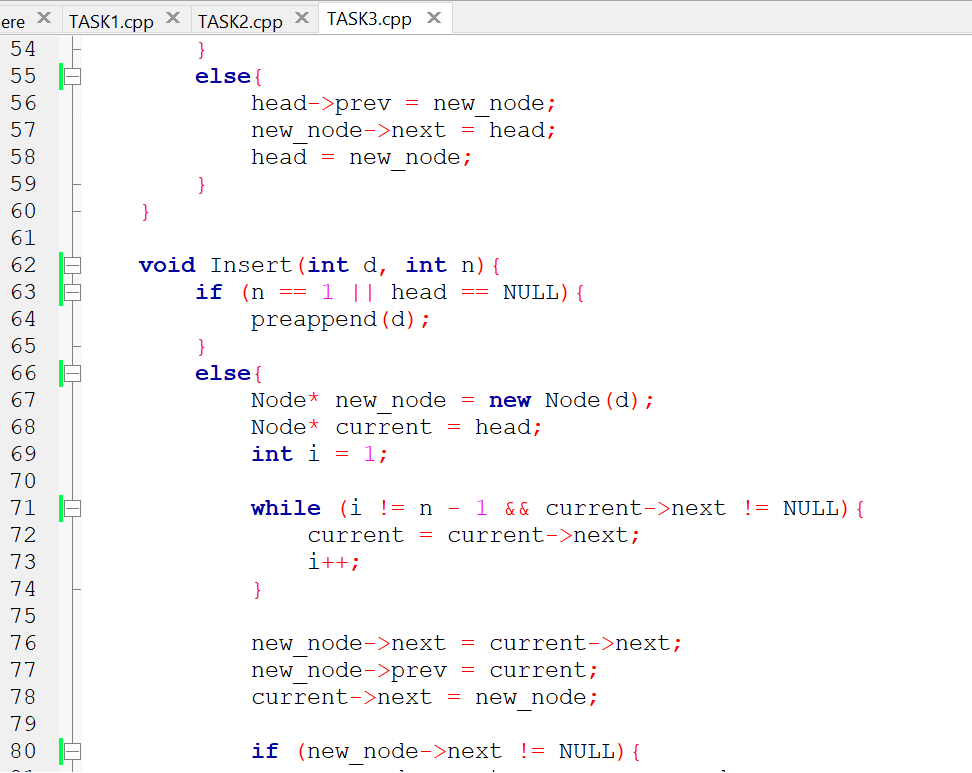
**TASK:3**

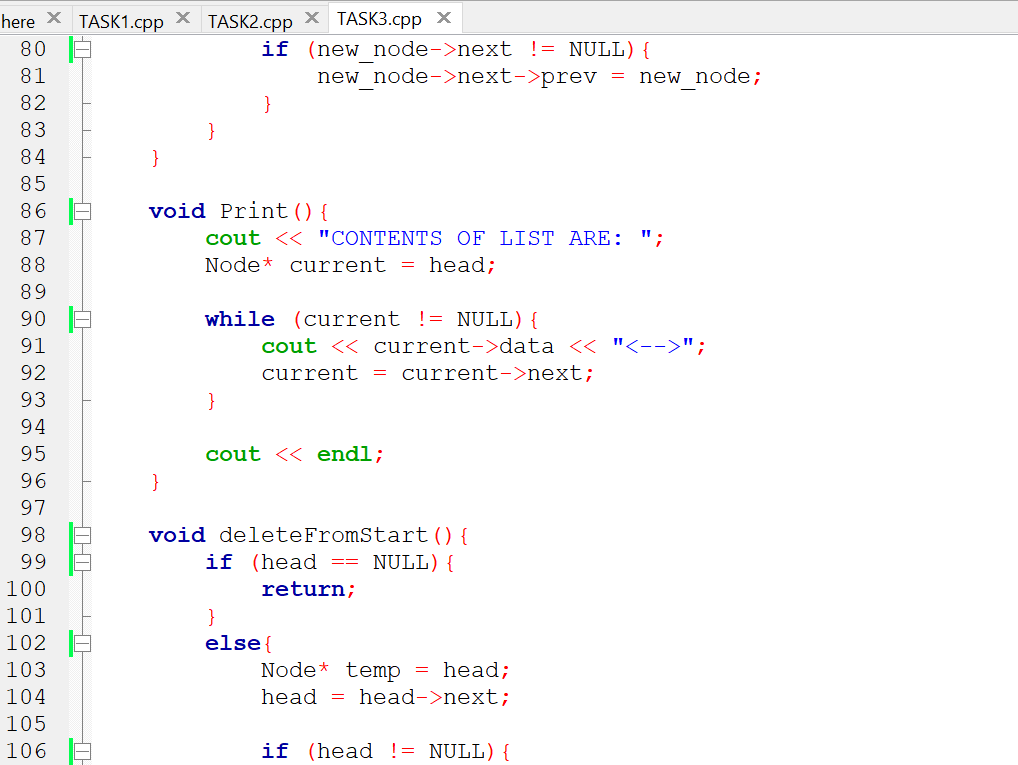
**CODE:**

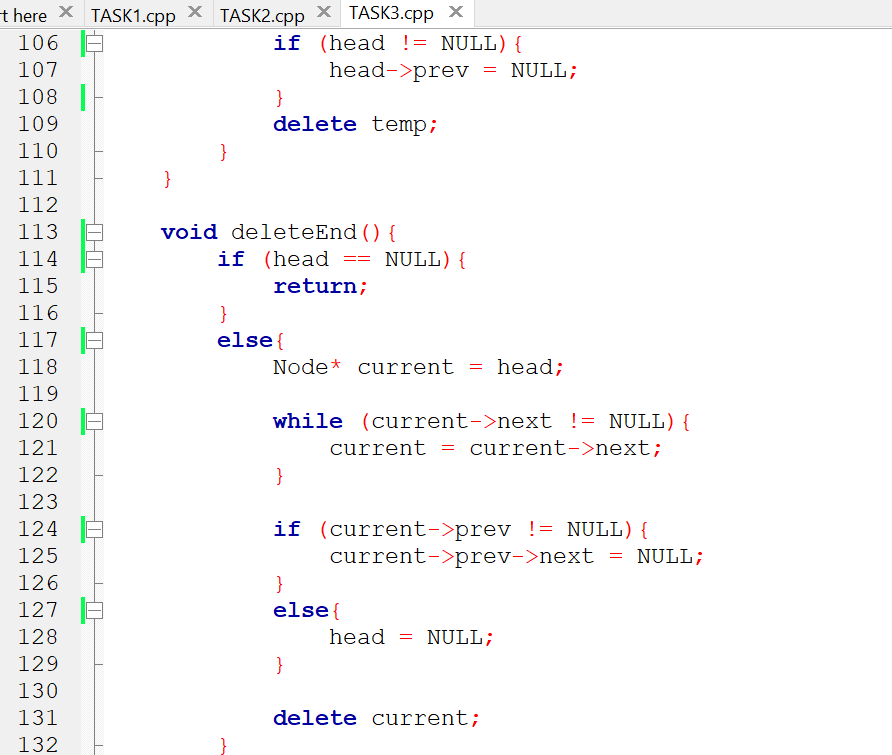
****

**A screenshot of a computer program

Description automatically generated with medium confidence**

****

****

****

**A screenshot of a computer program

Description automatically generated with medium confidence**

**A screenshot of a computer program

Description automatically generated with low confidence**

**A screenshot of a computer program

Description automatically generated with medium confidence**

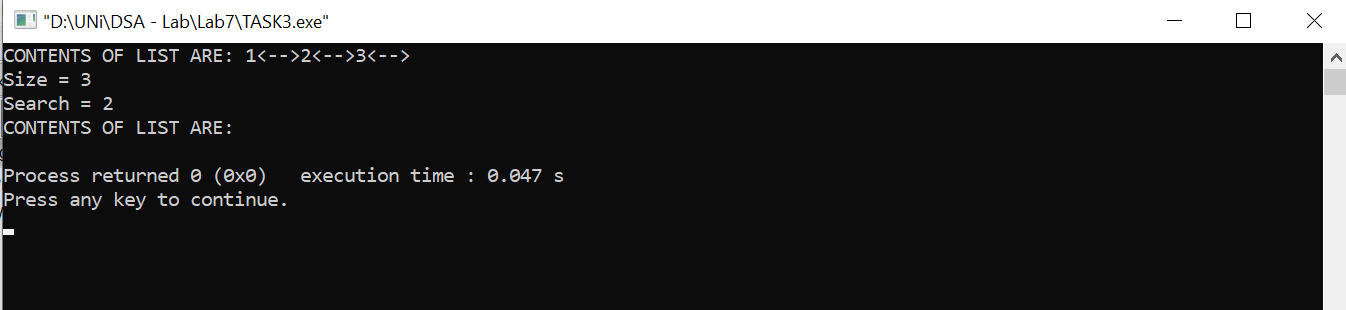
**A screen shot of a computer

Description automatically generated with medium confidence**

**A screenshot of a computer screen

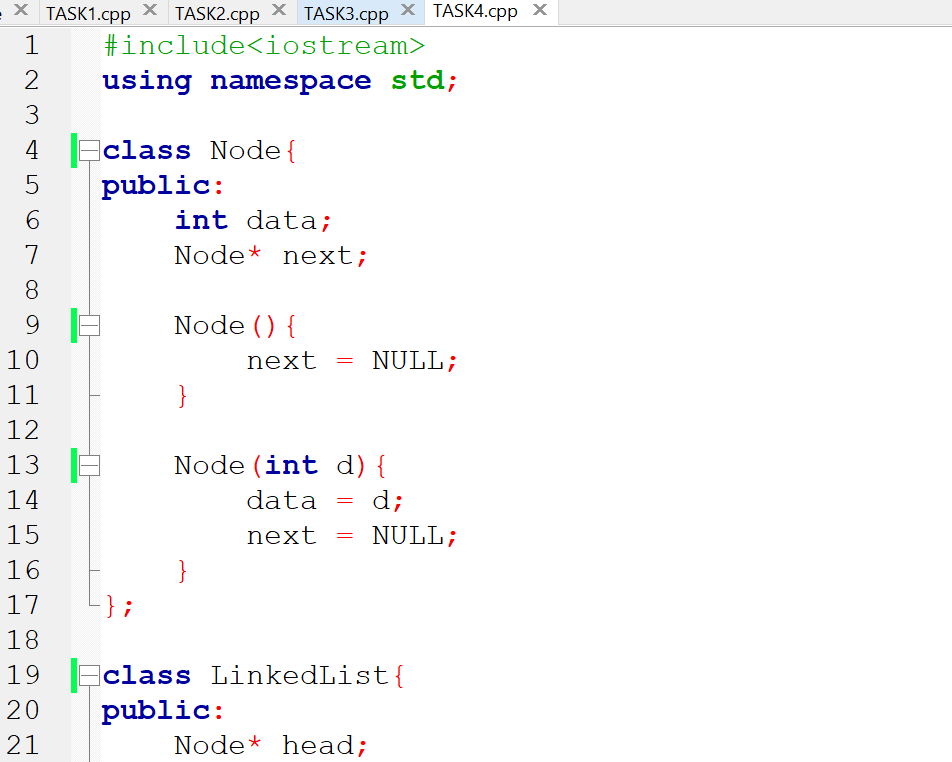
Description automatically generated with low confidence**

**OUTPUT:**



**TASK:4**

**CODE:**



A picture containing text, screenshot, number, font

Description automatically generated

A screen shot of a computer code

Description automatically generated with low confidence

A screenshot of a computer code

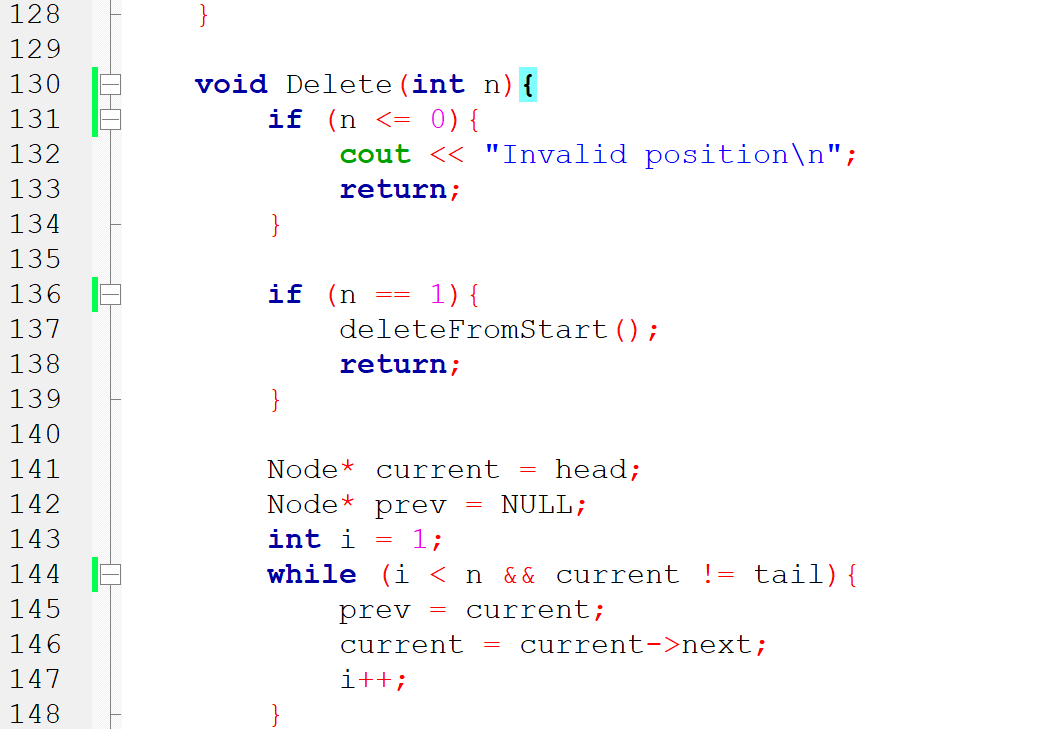
Description automatically generated with low confidence

A screen shot of a computer code

Description automatically generated with low confidence

A screen shot of a computer code

Description automatically generated with medium confidence



A screen shot of a computer code

Description automatically generated with low confidence

A screenshot of a computer code

Description automatically generated with medium confidenceA screen shot of a computer code

Description automatically generated with low confidence

A screen shot of a computer code

Description automatically generated with low confidence

A screenshot of a computer code

Description automatically generated with medium confidence

A screenshot of a computer code

Description automatically generated with low confidence

A screenshot of a computer code

Description automatically generated with medium confidence

**OUTPUT:**

A picture containing text, software, multimedia software, screenshot

Description automatically generated