# Laboratory 8

1. Questions
   1. Implement a linked list and perform following operations.
      1. Insert a node before and after a given node
      2. Delete a node before and after a given node
   2. Implement a linked list to create and print a binary tree.
2. Algorithm

**2.1 Insert a node before and after a given node-Delete a node before and after a given node**

Step 1: start

Step 2: to add node before the given node

2.1 allocate node

2.2 put in the data

2.3 check if head == null

2.4 if it is, head = new and head->next= NULL

2.5 else, iterate through the nodes till found key

2.6 point the previous node to the newnode

2.7 point the newnode to the next node

Step 3: to add node after the given node

3.1 allocate node

3.2 put in the data

3.3 check if head == null

3.4 if it is, head = new and head->next= NULL

3.5 else, iterate through the nodes till found key

3.6 point the key node to the newnode

3.7 point the newnode to the next node

Step 4: to delete node before the given node

4.1 iterate through the nodes till found key

4.2 then, prev->next = temp->next

4.3 free(temp)

Step 5: to delete node after the given node

4.1 iterate through the nodes till found key

4.2 del= temp->next

4.2 then, temp->next = del->next

4.3 free(del)

Step 6: stop

**2.2 using linked list , create and print a binary tree.**

Step 1: start

Step 2: allocate node

Step 3: put in the data

Step 4: push data into a linked list

4.1 if (head == NULL):

4.2 head = temp;

4.3 head->next = NULL

4.4 else: temp->next= head and head = temp

Step 5: print the linked list

Step 6: allocate a newtreenode

6.1 temp ->info=value

6.2 emp->count = 0;

6.3 temp->left= temp->right = NULL;

Step 7: insert function

7.1 if root=NULL: return newtreenode(key)

7.2 if root->left = NULL: root->left = newTreeNode(key);

7.3 if root->right = NULL: root->right = newTreeNode(key);

7.4 if (temp->count! =2): insert(root->left,key);

else {

temp = root->right;

7.4.1 if (temp->count! =2)

insert(root->right,key);

7.4.2 else

insert(root->left,key);

7.5 return root

Step 8: stop

1. Program

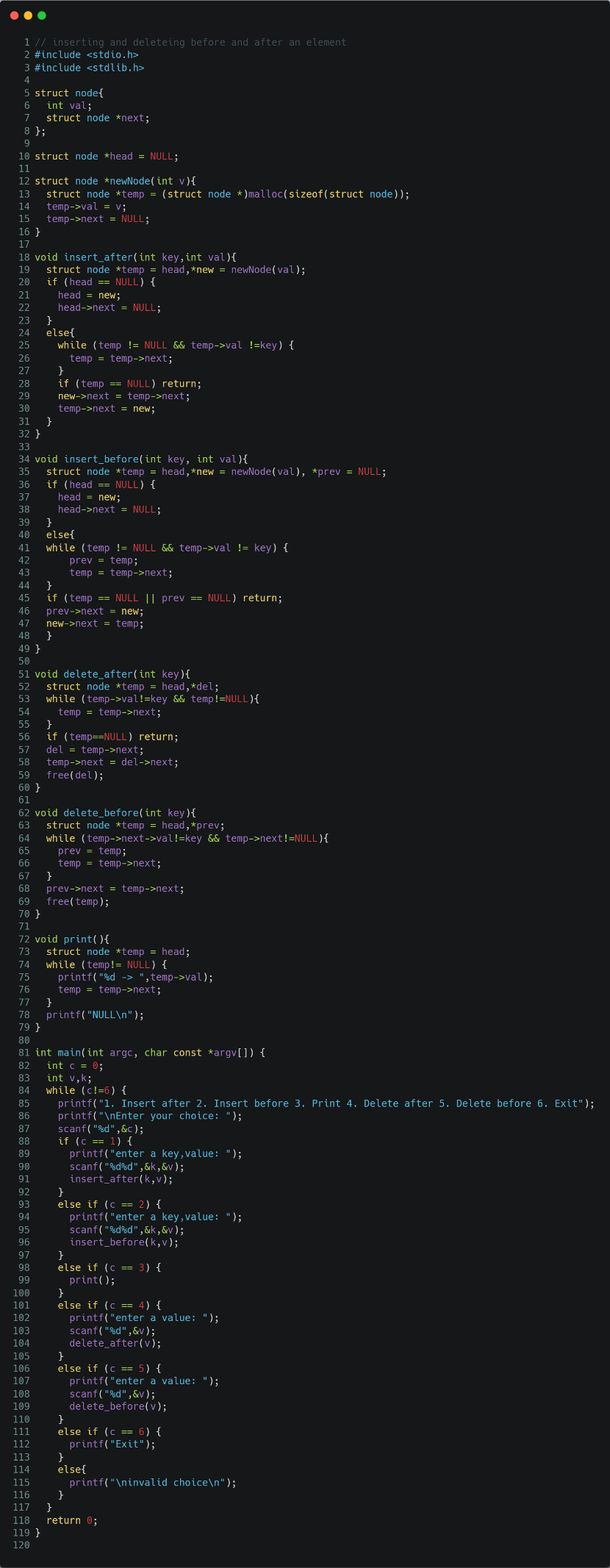


Figure program to Insert a node before and after a given node-Delete a node before and after a given node

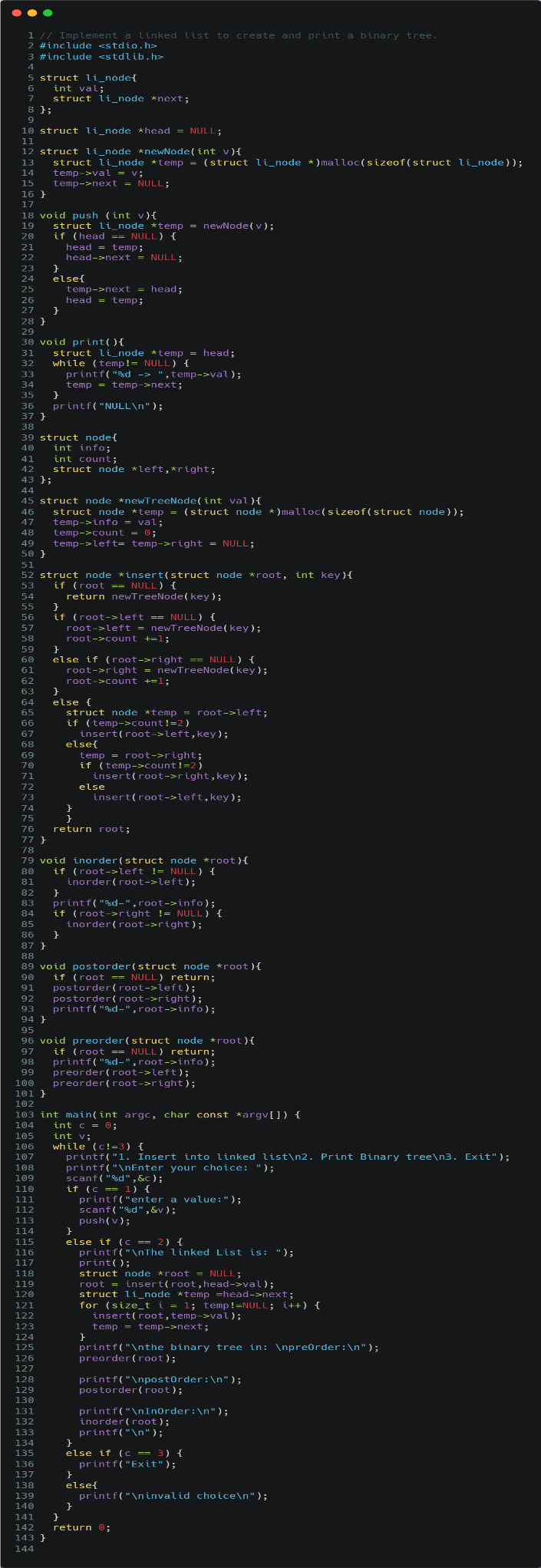


Figure program to create and print a binary tree using linked list

1. Presentation of Results

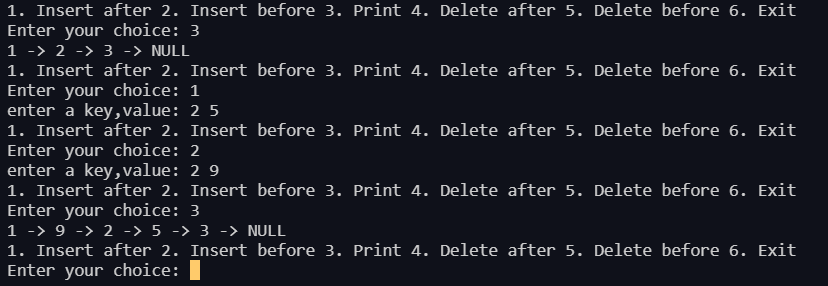


Figure inserting a node before and after a given node

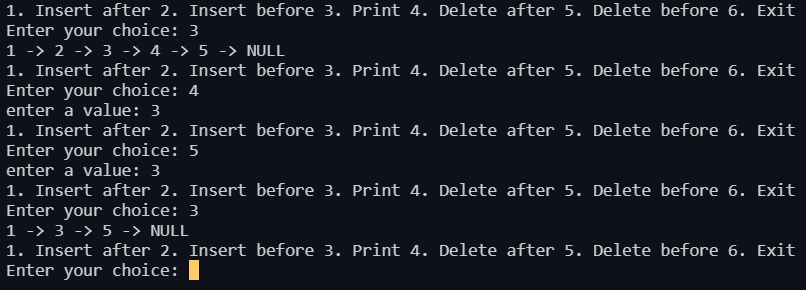


Figure deleting a node before and after a given node

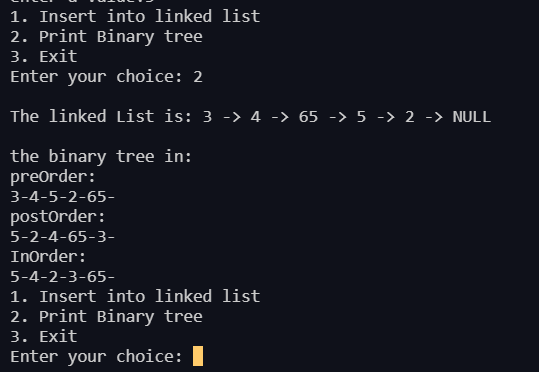


Figure output of program to create and print a binary tree using linked list

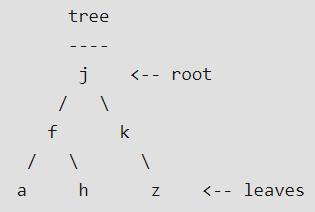
1. Conclusions

Learning happened

**Trees** are hierarchical data structures.

The topmost node is called **Root** of the tree. The elements that are directly under an element are called **its Children**. The element directly above something is called its **Parent**.

**Binary Tree**: A tree whose elements have at most 2 children is called a binary tree. Since each element in a binary tree can have only 2 children, we typically name them the left and right child.

**Example**:

Here,

* **‘j’** is the root.
* children of **‘f’** are **‘a’** and **‘h’**
* Parent of **‘a’** and **‘h’** is **‘f’**
* **‘a’**, **‘h’** and **‘z’** are leaves of the tree.