Experiment No. 5

Name : Manjiri Chavande	**DSE SY COMPS**
Division: B batch-A	**UID: 2023301003**

• Problem Statement

Construct binary tree from postorder and inorder traversal

• Theory

A Binary tree is represented by a pointer to the topmost node (commonly known as the “root”) of the tree. If the tree is empty, then the value of the root is NULL. Each node of a Binary Tree contains the following parts:

1. Data
2. Pointer to left child
3. Pointer to right child

Basic Operation On Binary Tree:

- Inserting an element.
- Removing an element.
- Searching for an element.
- Traversing the tree.

• Algorithm

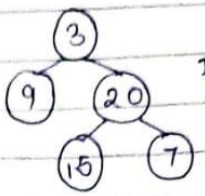
1. Take index’s the root element (in postorder traversal it starts from the last position i.e: n-1)
2. Find the position of the root in inorder
3. Root->right (position+1 to inorder’s end)
4. Root->left (inorder’s start to position-1)

Note : In case of posorder and inorder we first make a recursive call to the right side of the node and then the left since the index pointer begins from the end of the postorder array

• Solution

Constructing Binary Tree from given postorder and Inorder Traversal

In → [9, 3, 15, 20, 7]
Post → [9, 15, 7, 20, 3]

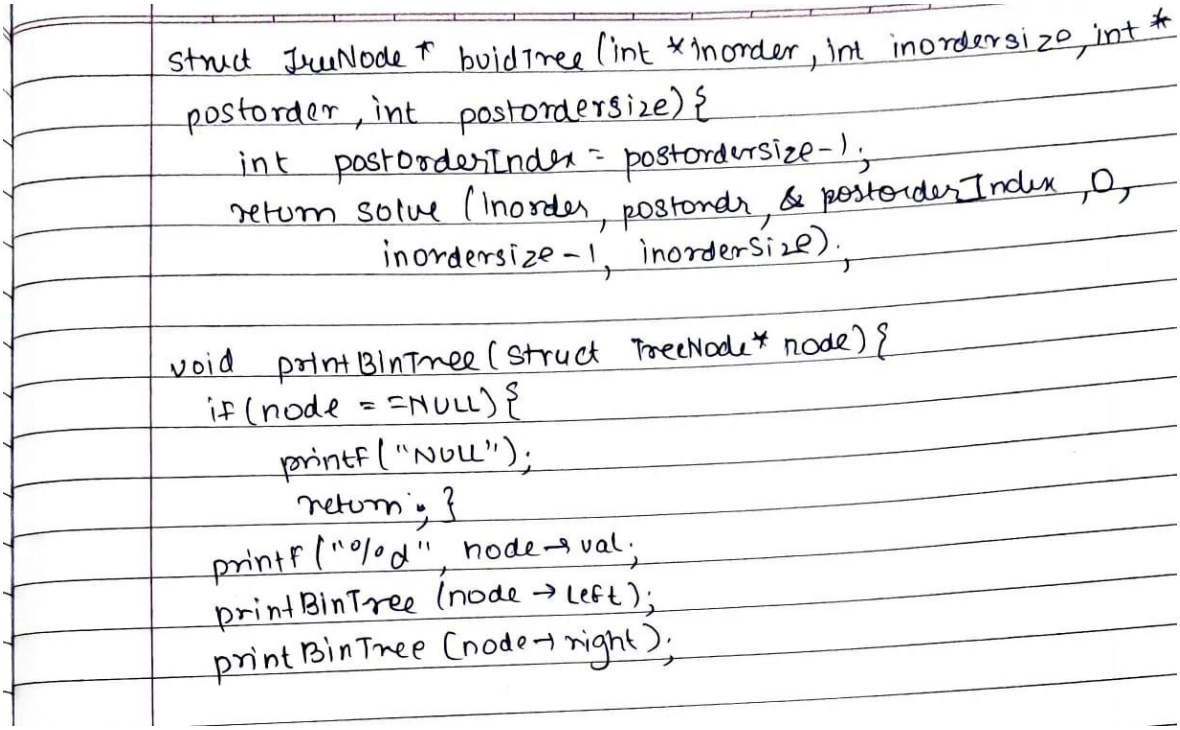


I-15, 20, 7
P- 9, 15, 7, 20

I - 15, 7
P: 7, 15

```
#include<stdlib.h>
#include<stdio.h>
struct TreeNode {
    int val;
    struct TreeNode *left;
    struct TreeNode *right;
};

struct TreeNode* createnode(int val){
struct TreeNode *node=(struct TreeNode*)
 malloc(sizeof(struct TreeNode));
node->val;
node->left;
node->right;
return node;
}

int FindPosition(int in[], int inorderSt
art, int inorderend, int element, int n){
for(int i = inorderstart; i<=inorderend;
    i++){ return i;
    }
 return -1;
}
```

```
struct TreeNode* solve(int
int post[], int *postorderIn
int inorderstart, int inorderE
 int n) {
if(*postorderIndex<0 ||
inorderstart>inorderEnd){
  return NULL;
  }

 int element = pos[(*postorder
   Index)--];
 struct TreeNode *root =
createNode(element);
int position = FindPosition(in
inorderStart, inorderEnd, elemen
n);

root->right = solve(in, post,
postorderIndex, position+1,
inorderEnd, n);
root->left = solve(in, post,
postorderIndex, inorderStart,
position-1, n);
 return root;
}
```



```
struct TreeNode * buildTree (int *inorder, int inordersize, int *
postorder, int postordersize){
    int postorderIndex = postordersize-1;
    return solve (inorder, postordr, & postorderIndex, 0,
            inordersize-1, inorderSize);

void printBinTree (struct TreeNode* node){
  if (node == NULL){
      printf ("NULL");
      return; }
printf ("%d", node->val);
printBinTree (node->left);
printBinTree (node->right);
```

- **Code**

```c
// Given two integer arrays postorder and inorder
// where postorder is the postorder traversal of a binary tree and
// inorder is the inorder traversal of the same tree, construct and return the
binary tree.

// Input: postorder = [9,15,7,20,3], inorder = [9,3,15,20,7]
// Output: [3,9,20,null,null,15,7]

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

  struct TreeNode {
    int val;
    struct TreeNode *left;
    struct TreeNode *right;
};

// Function to create a new node
struct TreeNode* create_node(int val) {
    struct TreeNode* node = (struct TreeNode*)malloc(sizeof(struct TreeNode));
    node->val = val;
    node->left = NULL;
    node->right = NULL;
```

```c
    return node;
}

// Function to find the position in the inorder array
int findPosition(int in[], int inorderStart, int inorderEnd, int element, int n)
{
    for (int i = inorderStart; i <= inorderEnd; i++) {
        if (in[i] == element) {
            return i;
        }
    }
    return -1;
}

// Function to build the binary tree
struct TreeNode* solve(int in[], int post[], int *postorderIndex, int
inorderStart, int inorderEnd, int n) {
    if (*postorderIndex < 0 || inorderStart > inorderEnd) {
        return NULL;
    }

    int element = post[(*postorderIndex)--];
    struct TreeNode* root = create_node(element);
    int position = findPosition(in, inorderStart, inorderEnd, element, n);

    // In case of inorder and postorder, we first build the right sub-tree
    root->right = solve(in, post, postorderIndex, position + 1, inorderEnd, n);
    root->left = solve(in, post, postorderIndex, inorderStart, position - 1, n);

    return root;
}

// Wrapper function to call the helper function
struct TreeNode* buildTree(int* inorder, int inorderSize, int* postorder, int
postorderSize) {
    int postorderIndex = postorderSize - 1;  // Start from the last index of
postorder
    return solve(inorder, postorder, &postorderIndex, 0, inorderSize - 1,
inorderSize);
}

// Function to print the preorder traversal
void printBinTree(struct TreeNode* node) {
    if (node == NULL) {
        printf("null ");
```

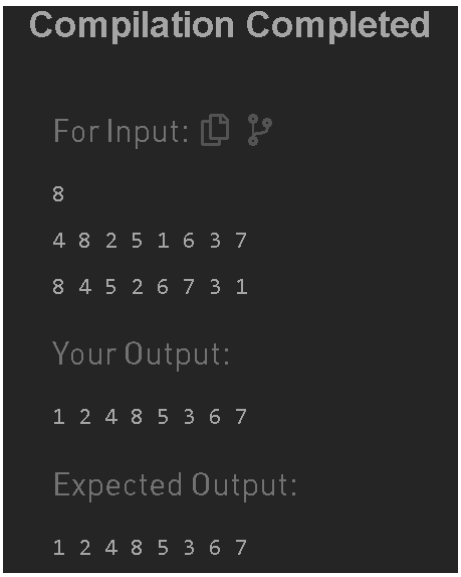
```
        return;
    }
    printf("%d ", node->val);
    printBinTree(node->left);
    printBinTree(node->right);
}
```

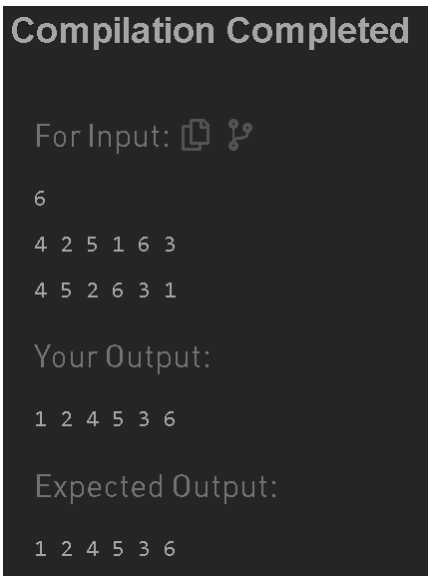
- **Output**

```
Constructed Binary Tree: 3 9 null null 20 15 null null 7 null null
PS D:\SY\DSA>
```

- **Test cases**

Without inclusion of **null**





- **Conclusion**

Thus we have successfully formed a binary tree from the given postorder and inorder.