**Given the following code that build a binary tree from an array. Use it to test the functions that you will build in this lab.**

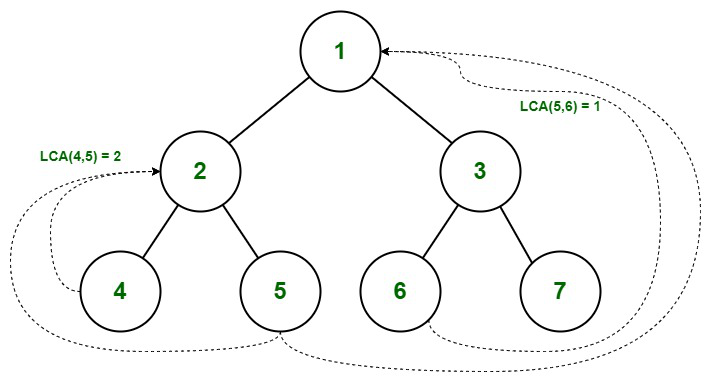
|  |
| --- |
| // Define the structure of a node  Typedef struct BTNode {      int data;      struct BTNode \*left;      struct BTNode \*right;  };  // Function to create a new node  struct BTNode\* createNode(int data) {      struct BTNode\* newNode = (struct Node\*)malloc(sizeof(struct BTNode));      newNode->data = data;      newNode->left = NULL;      newNode->right = NULL;      return newNode;  }  // Function to build a binary tree  struct BTNode\* buildTree(int arr[], int index, int size) {      struct BTNode\* root = NULL;      // Base case for recursion      if (index < size) {          // Create a new node with the current element          root = createNode(arr[index]);          // Recursively construct the left and right subtrees          root->left = buildTree(arr, 2 \* index + 1, size);          root->right = buildTree(arr, 2 \* index + 2, size);      }      return root;  }  // Function to print the inorder traversal of the binary tree  void inorderTraversal(struct BTNode\* root) {      if (root != NULL) {          inorderTraversal(root->left);          printf("%d ", root->data);          inorderTraversal(root->right);      }  }  int main() {      int arr[] = {1, 2, 3, 4, 5, 6, 7}; // Example array representing the binary tree      int size = sizeof(arr) / sizeof(arr[0]);      // Build the binary tree      struct BTNode\* root = buildTree(arr, 0, size);      // Print the inorder traversal of the binary tree      printf("Inorder Traversal: ");      inorderTraversal(root);      printf("\n");      return 0;  } |

1. *Complete the following recursive function aiming to print the values in a BT that are outside the range [a, b].*

|  |
| --- |
| void printOutsideRange(BTNode \*root, int a, int b) {  if (root == NULL)  return;  //Complete code here  } |

1. *Complete the following recursive function aiming to check if a binary tree is a sum tree. A binary sum tree is a binary tree where the value of a node is equal to the sum of the nodes present in its left subtree and right sub-tree. You should provide a solution with complexity O(n).*

|  |
| --- |
| int isSumTree(BTNode \*root) {  int isSumT = 1;  isSumTreeUtil(root, &isSumT);  return isSumT;  }  int isSumTreeUtil(BTNode \*root, int \*isSumT) {  if (root == NULL)  return 0;  //Complete code here  } |

1. *In this exercise, we want to implement the lowest common ancestor problem in binary trees. The***lowest common ancestor***is the lowest node in the tree that has both n1 and n2 as***descendants,***where n1 and n2 are the nodes for which we wish to find the LCA. Hence, the LCA of a binary tree with nodes n1 and n2 is the shared ancestor of n1 and n2 that is located farthest from the root. In the figure on the right, the LCA of the nodes 4 and 5 is 2. The LCA of nodes 6 and 5 is 1.*
2. *Given a Binary Tree having positive and negative values, write a function that prints the sum of values at each level and returns the highest one.*

|  |
| --- |
| Input : 4  / \  2 -5  / \ /\  -1 3 -2 6  Output: 6  Explanation :  Sum of all nodes of 0'th level is 4  Sum of all nodes of 1'th level is -3  Sum of all nodes of 0'th level is 6  Hence maximum sum is 6  Input : 1  / \  2 3  / \ \  4 5 8  / \  6 7  Output : 17 |