**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;  }  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);      return 0;  } |

1. *Write a function sumTree that returns the sum of all values in a binary tree.*

|  |
| --- |
| int sumTree(BTree root) {  // insert code here  } |

1. *Write a function countLeaves that returns the number of leaf nodes (nodes with no children).*

|  |
| --- |
| int countLeaves(BTree root) {  // insert code here  } |

1. *Write a function countFullNodes that returns the number of nodes with****both left and right children****.*

|  |
| --- |
| int countFullNodes(BTree root) {  // insert code here  } |

1. *Write a function fillArrayInOrder that fills an array with the tree’s values using****in-order traversal****.*

|  |
| --- |
| void fillArrayInOrderUtil(BTree root, int arr[], int \*index) {  // insert code here  }  void fillArrayInOrder(BTree root, int arr[]){  // insert code here  } |

1. *Write a function maxMinDifference that returns the difference between the****maximum****and****minimum****values in the tree.*

|  |
| --- |
| // Helper functions to find max and min  int findMax(BTree root) {  // insert code here  }  int findMin(BTree root) {  // insert code here  }  int maxMinDifference(BTree root) {  // insert code here  } |