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);
    }
}
```

```c
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. Complétez la fonction récursive suivante qui doit imprimer les valeurs dans un arbre binaire qui sont hors l'intervalle [a, b].

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

```c
void printOutsideRange(BTNode *root, int a, int b) {
   if (root == NULL)
      return;
   //Complete code here




}
```

2. Complétez la fonction récursive suivante visant à vérifier si un arbre binaire est un arbre somme. Un arbre binaire somme est un arbre binaire où la valeur d'un noeud est égale à la somme des noeuds présents dans son sous-arbre gauche et son sous-arbre droit. Vous devez fournir une solution de complexité O(n).

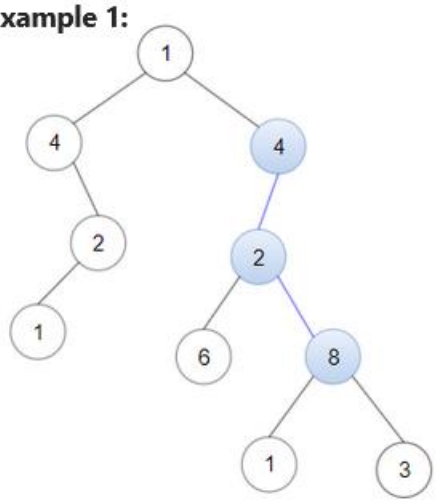
   Complete the following recursive function aiming to check if a binary tree is a sum tree. A binary sum tree is a binary tree where the value of a node is equal to the sum of the nodes present in its left subtree and right sub-tree. You should provide a solution with complexity O(n).

```c
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

}
```

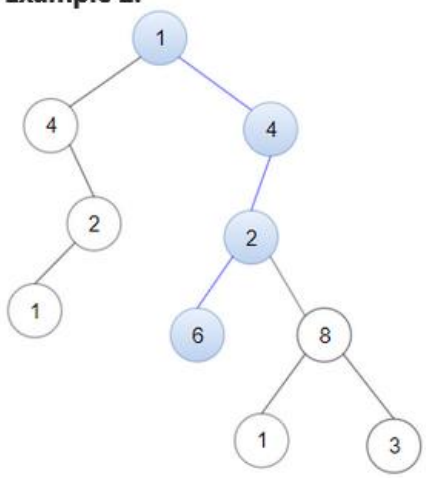
3. Étant donné un arbre binaire et une liste chaînée, compléter la fonction qui renvoie 1 si tous les éléments de la liste chaînée, en commençant par la tête, correspondent à un chemin descendant connecté dans l'arbre binaire, sinon elle renvoie 0. Voici deux exemples :
   Given a binary tree root and a linked list with head as the first node. We want to complete the function that returns 1 if all the elements in the linked list starting from the head correspond to some downward path connected in the binary tree otherwise return 0. In this context downward path means a path that starts at some node and goes downwards. Here is two examples:

Example 1:



The list 4 -> 2 ->8 exists in the above tree.

Example 2:



The list 1 -> 4 -> 2 -> 6 exists in the above tree.

Hint: The given problem can be solved with the help of the DFS Traversal of the Binary tree. During the DFS traversal, if any node of the Binary Tree is equal to the head of the linked list, a recursive function **isPathUntil()** can be called to recursively check whether the other elements of the linked list also exist as a path from that node. If the complete linked list has been traversed, there exists a valid required path, hence return **true**. Otherwise, return **false**.

```
typedef struct ListNode{
     element data;
     struct ListNode *next;
} ListNode;
```

```
bool isPathUtil(struct ListNode* curList, struct TreeNode* curTree)
{
    //Complete code here
}

// Function to check if the linked list
// exist as a downward path in Binary tree
// using the DFS Traversal of the Tree
bool isSubPath(struct ListNode* head, struct TreeNode* root){
 //complete code here
}
```