```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

struct Node {
    int data;
    Node* left, *right;
    Node(int x) {
        data = x;
        left = right = NULL;
    }
};

void inorderTraversal(Node* root, vector<int>& arr) {
    if(root == NULL)
        return;
    inorderTraversal(root->left, arr);
    arr.push_back(root->data);
    inorderTraversal(root->right, arr);
}

void createBST(Node* root, vector<int>& arr, int& idx) {
    if(root == NULL)
        return;
    createBST(root->left, arr, idx);
    root->data = arr[idx++];
    createBST(root->right, arr, idx);
}

Node* convertToBST(Node* root) {
    vector<int> arr;
    inorderTraversal(root, arr);
    sort(arr.begin(), arr.end());
    int idx = 0;
    createBST(root, arr, idx);
    return root;
}

void eulerTree(struct Node* root, vector<int> &Euler)
{
    // store current node's data
    Euler.push_back(root->data);

    // If left node exists
    if (root->left)
    {
        // traverse left subtree
        eulerTree(root->left, Euler);

        // store parent node's data
        Euler.push_back(root->data);
    }

    // If right node exists
    if (root->right)
    {
```

```
        // traverse right subtree
        eulerTree(root->right, Euler);

        // store parent node's data
        Euler.push_back(root->data);
    }
}


int printEulerTour(Node *root)
{
    // Stores Euler Tour
    vector<int> Euler;

    eulerTree(root, Euler);
    int sum = 0;
    for (int i = 0; i < Euler.size(); i++) {
        // cout << Euler[i] << " ";
        sum += Euler[i];
    }
    return sum;
}


int main() {
    int n;
    cin >> n;
    vector<Node*> nodes(n);
    for(int i=0; i<n; i++) {
        int val;
        cin >> val;
        nodes[i] = new Node(val);
    }
    for(int i=1; i<n; i++) {
        int parentIdx, childIdx;
        cin >> parentIdx >> childIdx;
        if(nodes[parentIdx-1]->left == NULL)
            nodes[parentIdx-1]->left = nodes[childIdx-1];
        else
            nodes[parentIdx-1]->right = nodes[childIdx-1];
    }
    Node* root = nodes[0];
    root = convertToBST(root);
    int points = 0;
    cout << printEulerTour(root);
    return 0;
}
```