Size in C++

```
#include <iostream>
#include <vector>
using namespace std;

// Node structure definition
struct Node {
    int data;
    vector<Node*> children;
};

// Function to display the tree structure
void display(Node* node) {
    cout << node->data << " -> ";
    for (Node* child : node->children) {
        cout << child->data << ", ";
    }
    cout << "." << endl;

    for (Node* child : node->children) {
        display(child);
    }
}

// Function to construct the tree from array representation
Node* construct(int arr[], int n) {
    Node* root = nullptr;
    vector<Node*> st;

    for (int i = 0; i < n; ++i) {
        if (arr[i] == -1) {
            st.pop_back();
        } else {
            Node* t = new Node();
            t->data = arr[i];

            if (!st.empty()) {
                st.back()->children.push_back(t);
            } else {
                root = t;
            }

            st.push_back(t);
        }
    }

    return root;
}

// Function to calculate the size of the tree
int size(Node* node) {
    int sz = 0;
    for (Node* child : node->children) {
        sz += size(child);
    }
    return 1 + sz;
}

int main() {
    // Static data representing the tree
    int arr[] = {10, 20, -1, 30, 50, -1, 60, -1, -1, 40, -1, -1};
```

Input Array:

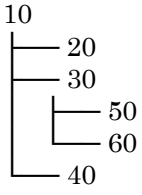
{10, 20, -1, 30, 50, -1, 60, -1, -1, 40, -1, -1}

🛠 Tree Construction Dry Run

This array uses -1 to indicate the end of children for a node. We construct the tree using a vector (acting like a stack).

Step	arr[i]	Stack Top	Action	Tree Changes
0	10	—	New Node(10), push	root = 10
1	20	10	Add 20 as child to 10, push	10 → 20
2	-1	20	Pop 20	
3	30	10	Add 30 as child to 10, push	10 → 30
4	50	30	Add 50 as child to 30, push	30 → 50
5	-1	50	Pop 50	
6	60	30	Add 60 as child to 30, push	30 → 60
7	-1	60	Pop 60	
8	-1	30	Pop 30	
9	40	10	Add 40 as child to 10, push	10 → 40
10	-1	40	Pop 40	
11	-1	10	Pop 10	Done

✅ Tree Structure:



Let's apply it:

- size(20) = 1
- size(50) = 1
- size(60) = 1
- size(30) = 1 (self) + size(50) + size(60) =

int n = sizeof(arr) / sizeof(arr[0]);
 	
// Construct the tree
Node* root = construct(arr, n);
 	
// Calculate the size of the tree
int sz = size(root);
cout << sz << endl; // Output should be 6
 	
// Display the tree structure (optional)
// display(root);
 	
return 0;
}
	1 + 1 + 1 = 3
	• size(40) = 1
	• size(10) = 1 (self) + size(20) + size(30) + size(40) = 1 + 1 + 3 + 1 = **6**

6	