Disjoint Set in C++

```
#include <bits/stdc++.h>
using namespace std;

vector<int> parent, rankVec; // Renamed rank to
rankVec

void makeSet(int n) {
    parent.resize(n + 1);
    rankVec.resize(n + 1, 0); // Use rankVec here
    for (int i = 0; i <= n; i++) {
        parent[i] = i;
    }
}

int findUPar(int node) {
    if (node == parent[node])
        return node;
    return parent[node] = findUPar(parent[node]);
}

void unionByRank(int u, int v) {
    int ulp_u = findUPar(u); // ultimate parent of u
    int ulp_v = findUPar(v); // ultimate parent of v
    if (ulp_u == ulp_v) return; // already in the same set

    // Union by rank
    if (rankVec[ulp_u] < rankVec[ulp_v]) { // Use
rankVec here
        parent[ulp_u] = ulp_v;
    }
    else if (rankVec[ulp_u] > rankVec[ulp_v]) { // Use
rankVec here
        parent[ulp_v] = ulp_u;
    }
    else {
        parent[ulp_v] = ulp_u;
        rankVec[ulp_u]++; // Use rankVec here
    }
}

int main() {
    int n = 7; // Number of elements
    makeSet(n);

    unionByRank(1, 2);
    unionByRank(2, 3);
    unionByRank(4, 5);
    unionByRank(6, 7);
    unionByRank(5, 6);

    // Check if 3 and 7 are in the same set
    if (findUPar(3) == findUPar(7)) {
        cout << "Same\n";
    } else {
        cout << "Not same\n";
    }

    unionByRank(3, 7);

    // Check again if 3 and 7 are in the same set
    if (findUPar(3) == findUPar(7)) {
```

Initial Setup

You're working with n = 7, i.e., elements from 1 to 7.

makeSet(n):

- parent[i] = i for all i ∈ [0, 7]
- rankVec[i] = 0 initially

✅ Union Operations

Step	Operation	Resulting Union	Parent Array	Rank Array (rankVec)
1	union(1, 2)	1 becomes parent of 2	[0, 1, 1, 3, 4, 5, 6, 7]	[0, 1, 0, 0, 0, 0, 0, 0]
2	union(2, 3)	1 becomes parent of 3 (via 2)	[0, 1, 1, 1, 4, 5, 6, 7]	[0, 1, 0, 0, 0, 0, 0, 0]
3	union(4, 5)	4 becomes parent of 5	[0, 1, 1, 1, 4, 4, 6, 7]	[0, 1, 0, 0, 1, 0, 0, 0]
4	union(6, 7)	6 becomes parent of 7	[0, 1, 1, 1, 4, 4, 6, 6]	[0, 1, 0, 0, 1, 0, 1, 0]
5	union(5, 6)	4 becomes parent of 6 (via 5)	[0, 1, 1, 1, 4, 4, 4, 6]	[0, 1, 0, 0, 2, 0, 1, 0]

❓ First Check: findUPar(3) vs findUPar(7)

- findUPar(3) → follows to 1
- findUPar(7) → 7 → 6 → 4
- So: **1 != 4** → Output: **Not same**

🔁 union(3, 7)

- Ultimate parents: 1 and 4
- Both have rank 2 → tie, choose one (say 1) as parent, and increment rank

Result	Updated Parent Array	Updated Rank Array
1 becomes parent of 4	[0, 1, 1, 1, 1, 4, 4, 6]	[0, 3, 0, 0, 2, 0, 1, 0]

❓ Second Check: findUPar(3) vs findUPar(7)

cout << "Same\n";
} else {
cout << "Not same\n";
}
 	
return 0;
}
	• findUPar(3) → 1
	• findUPar(7) → 7 → 6 → 4 → 1
	• So: **1 == 1** → Output: **Same**
	
	✅ **Final Output**
	
	Not same
	Same

Output:- Not same Same	