Union-Find data structure

Problem:

There are *n* computers. Your job is to connect them so that you can send data from one computer to other. We perform two operations on them:

- `Connect(A,B)` connects computers *A* and *B* if they are not in the same component.
- `Connected(A,B)?` is **TRUE** if *A* and *B* are in the same component, and **FALSE** otherwise.

Your job is to print "T" if `Connected(A, B)?` returns True, or "F" otherwise. See sample input/ output for clarification.

Input:

The first line of the input contains an integer *n*, where *n* denotes the initial number of computers you need to connect. Each of the next lines is an operation. Each operation has the following format:

```
Operation-type      A     B
```

Where, 1 <= A, B <= n

Operation-type	Arguments	Meaning
0	A B	`Connect(A, B)`
1	A B	`Connected(A, B)?`

Input ends with the line: **-1 -1 -1** and you don't need to process this line.

Sample input/output:

Sample Input	Sample output
5 0 1 2 1 1 3 1 2 1 0 3 4 0 4 5 1 3 5 0 1 4 1 1 5 -1 -1 -1	F T T T

A possible implementation of union-find:

```
class UnionFind
{
    // data

    // methods
    public:
    // Constructor
    UnionFind(int size);

    // Destructor
    ~UnionFind();


    // Find operation
    int Find(int x);

    // Union operation
    void Union(int x, int y);

    // Connected(x, y)?
    bool Connected(int x, int y);

    // additional methods if needed

};
```

Input processing:

```
/*
      Author: Doe
      Date: January 15, 2015
      Description: An implementation of Union-Find data structure
      Implementation: It implements X, Y, and Z
      Complexity:
*/

int main(void)
{
    int n;
    int op, x, y;

    //freopen("uf-medium.in", "r", stdin);

    scanf("%d", &n);

    UnionFind *uf = new UnionFind(n);
    while (3 == scanf("%d %d %d", &op, &x, &y))
    {
        //printf("%d %d %d\n", op, x, y);

        if (op == -1 && x == -1 && y == -1) break;

        if (!op)    // Connect/ Union
        {
               …
        }
        else        // Connected/ Find
        {
               …
        }
    }
    uf->~UnionFind();

    return 0;
}
```

Grading rubric:

Criteria	Points
Code compiles without any error	3
Code gives correct answers in test cases	4
Code is properly documented (e.g., comments)	2
Implements Quick Union and Quick Find	0.5 + 0.5
Total	10