```cpp
#include "bintree.h"
#include <iostream>

#include <cmath>
#include <queue>
#include <iomanip>

using namespace std;

template <class T>
bool Similares(const bintree<T> &A1, const typename bintree<T>::node &v1, const bintree<T> &A2, const typename bintree<T>::node &v2)
{
    if ((v1.null() && !v2.null()) || (!v1.null() && v2.null()))
    {
        return false;
    }
    if (!v1.null() && !v2.null())
    {
        if(!Similares(A1, v1.left(), A2, v2.left()))
        {
            return false;
        }
        if(!Similares(A1, v1.right(), A2, v2.right()))
        {
            return false;
        }
    }
    return true;

}

template <class T>
void MostrarArbol(const bintree<T> &A, typename bintree<T>::node root)
{
    queue<typename bintree<T>::node> colaNodos;
    int totalNodos = A.size();
    int techo = log2(totalNodos + 1);
    colaNodos.push(root);
    int pot = 0;
    while (colaNodos.size() > 0)
    {
        int niveles = colaNodos.size();
        while (niveles > 0)
        {
            typename bintree<T>::node nodoAux = colaNodos.front();
            colaNodos.pop();
            cout << setw((niveles == pow(2, pot)) ? pow(2, (techo - pot)) : pow(2, (techo - pot + 1)));
            cout << *nodoAux;
            if (!nodoAux.left().null())
                colaNodos.push(nodoAux.left());
            if (!nodoAux.right().null())
                colaNodos.push(nodoAux.right());
            niveles--;
        }
        pot++;
        cout << endl;
    }
}
```

```
59
60 int main()
61 {
62     //ARBOL1
63     bintree<int> arb1(0);
64     arb1.insert_left(arb1.root(), 1);
65     arb1.insert_right(arb1.root(), 2);
66
67
68     bintree<int>::node aux1 = arb1.root().left();
69     arb1.insert_left(aux1, 3);
70     arb1.insert_right(aux1, 4);
71
72     aux1 = arb1.root().right();
73     arb1.insert_left(aux1, 5);
74     arb1.insert_right(aux1, 6);
75
76     //ARBOL 2
77     bintree<int> arb2(0);
78     arb2.insert_left(arb2.root(), 1);
79     arb2.insert_right(arb2.root(), 2);
80
81     bintree<int>::node aux2 = arb2.root().left();
82     arb2.insert_left(aux2, 3);
83     arb2.insert_right(aux2, 4);
84
85     aux2 = arb2.root().right();
86     arb2.insert_left(aux2, 5);
87     arb2.insert_right(aux2, 6);
88
89     //ARBOL 3
90     bintree<int> arb3(0);
91     arb3.insert_left(arb3.root(), 1);
92     arb3.insert_right(arb3.root(), 2);
93
94     bintree<int>::node aux3 = arb3.root().left();
95     arb3.insert_left(aux3, 3);
96     arb3.insert_right(aux3, 4);
97
98     aux3 = arb3.root().right();
99     arb3.insert_left(aux3, 5);
100    arb3.insert_left(aux3.left(),8);
101
102    cout << "----------ARBOL ORIGINAL----------" << endl;
103    MostrarArbol(arb1, arb1.root());
104    cout << "----------ARBOL ORIGINAL----------" << endl;
105    MostrarArbol(arb2, arb2.root());
106    cout << "----------ARBOL ORIGINAL----------" << endl;
107    MostrarArbol(arb3, arb3.root());
108
109    if (Similares(arb1, arb1.root(), arb3, arb3.root()))
110    {
111        cout << "El arbol 1 y el 3 son similares " << endl;
112    }
113    else
114    {
115        cout << "El arbol 1 y 3 no son similares " << endl;
116    }
117
118    if (Similares(arb1, arb1.root(), arb2, arb2.root()))
```

```
119     {
120         cout << "El arbol 1 y el 2 son similares " << endl;
121     }
122     else
123     {
124         cout << "El arbol 1 y 2 no son similares " << endl;
125     }
126 }
```