```cpp
#include "bintree.h"

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

#include <iostream>

using namespace std;

template <class T>
bool esHoja(const bintree<T> &A, const typename bintree<T>::node &v)
{
    return ( v.left().null() && v.right().null() );
}

template <class T>
int profundidad(const bintree<T> &A, const typename bintree<T>::node &v)
{
    int prof=0;
    typename bintree<T>::node aux=v;
    while(A.root()!=aux){
        prof++;
        aux=aux.parent();
    }
    return prof;
}

template <class T>
int getMaxProfundidad(const bintree<T> &A, typename bintree<T>::node v){
    static int max_profundidad=0;

    if(v==A.root()) //ES DECIR HEMOS LLAMADO DE NUEVO A LA FUNCION CON OTRO ARBOL
    {
        max_profundidad=0;
    } //HAY QUE RESETEAR LOS DATOS

    if(!v.null())
    {
        if(esHoja(A,v))
        {
            int prof=profundidad(A,v);
            if(prof>max_profundidad)
            {
                max_profundidad=prof;
            }
        }
        getMaxProfundidad(A,v.left());
        getMaxProfundidad(A,v.right());
    }
    return max_profundidad;
}

int getNodosCompletos(int prof)
{
    static int sum=0;
    sum+=pow(2,prof);
    prof--;
    if(prof<0)
    {
```

```
61          int aux=sum;
62          sum=0;
63          return aux;
64      }else
65      {
66          getNodosCompletos(prof);
67      }
68
69  }
70
71  template <class T>
72  bool esCompleto(const bintree<T> &A)
73  {
74      return getNodosCompletos(getMaxProfundidad(A,A.root())) == A.size();
75  }
76
77  template <class T>
78  void MostrarArbol(const bintree<T> &A,typename bintree<T>::node root){
79      queue<typename bintree<T>::node> colaNodos;
80      int totalNodos=A.size();
81      int techo=log2(totalNodos+1);
82      colaNodos.push(root);
83      int pot=0;
84      while(colaNodos.size() > 0){
85              int niveles = colaNodos.size();
86              while(niveles > 0){
87                  typename bintree<T>::node nodoAux = colaNodos.front();
88                  colaNodos.pop();
89                  cout<<setw((niveles==pow(2,pot))?pow(2, (techo-pot)):pow(2, (techo-
    pot+1)));
90                  cout<<*nodoAux;
91                  if(!nodoAux.left().null()) colaNodos.push(nodoAux.left());
92                  if(!nodoAux.right().null()) colaNodos.push(nodoAux.right());
93                  niveles--;
94              }
95              pot++;
96              cout << endl;
97      }
98  }
99
100 int main()
101 {
102     bintree<int> arb(0);
103     arb.insert_left(arb.root(),1);
104     arb.insert_right(arb.root(),2);
105
106     bintree<int>::node aux = arb.root().left();
107     arb.insert_left(aux,3);
108     arb.insert_right(aux,4);
109
110     aux = arb.root().right();
111     arb.insert_left(aux,5);
112     arb.insert_right(aux,6);
113
114     MostrarArbol(arb,arb.root());
115
116     if(esCompleto(arb))
117     {
118         cout << "El arbol binario es completo " << endl;
119     }
```

```
120     else
121     {
122         cout << "El arbol binario no es completo" << endl;
123     }
124 
125 }
```