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


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

using namespace std;


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;
    }
}


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>
using stack_node = stack<typename bintree<T>::node>;

template <class T>
```

```
60 int Altura(const bintree<T> &A)
61 {
62     stack_node<T> pila;
63     pila.push(A.root());
64     typename bintree<T>::node aux;
65     int max_profun = 0;
66     while (!pila.empty())
67     {
68         aux = pila.top();
69         pila.pop();
70         cout << *aux ; //Para realizar pruebas he puesto este cout aqui en medio
71         if(esHoja(A,aux))
72         {
73             int prof=profundidad(A,aux);
74             if(prof>max_profun)
75             {
76                 max_profun=prof;
77             }
78         }
79         if (!aux.right().null())
80             pila.push(aux.right());
81         if (!aux.left().null())
82             pila.push(aux.left());
83     }
84     return max_profun;
85 }
86
87 template <class T>
88 void postorden(const bintree<T> &A, const typename bintree<T>::node &v)
89 {
90     if (!v.null())
91     {
92         cout << *v;
93         postorden(A,v.left());
94         postorden(A,v.right());
95     }
96 }
97
98
99 int main()
100 {
101     bintree<int> arb(0);
102     arb.insert_left(arb.root(),1);
103     arb.insert_right(arb.root(),2);
104
105     bintree<int>::node aux = arb.root().left();
106     arb.insert_left(aux,3);
107     arb.insert_right(aux,4);
108
109     aux = arb.root().right();
110     arb.insert_left(aux,5);
111     arb.insert_right(aux,6);
112
113     cout << "----------ARBOL ORIGINAL----------" << endl;
114     MostrarArbol(arb,arb.root());
115     cout << "---------PRUEBA RECURSIVA---------" << endl;
116     postorden(arb,arb.root());
117     cout << endl;
118     cout << "---------PRUEBA CON PILA----------" << endl;
119     int h= Altura(arb);
```

```
120     cout << endl;
121     cout << "La altura del arbol es -> " << h << endl;
122 
123 }
```