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

#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>
void reflexion(bintree<T> &A, typename bintree<T>::node v)
{
    if (!v.null())
    {
        bintree<T> aux1;
        bintree<T> aux2;
        A.prune_right(v,aux1);
        A.prune_left(v,aux2);
        A.insert_left(v,aux1);
        A.insert_right(v,aux2);
        reflexion(A,v.right());
        reflexion(A,v.left());

    }
}

int main()
{
    bintree<int> arb(0);
    arb.insert_left(arb.root(),1);
    arb.insert_right(arb.root(),2);

    bintree<int>::node aux = arb.root().left();
    arb.insert_left(aux,3);
    arb.insert_right(aux,4);
```

```
60
61      aux = arb.root().right();
62      arb.insert_left(aux,5);
63      arb.insert_right(aux,6);
64
65      cout << "----------ARBOL ORIGINAL----------" << endl;
66      MostrarArbol(arb,arb.root());
67
68      cout << "-----------ARBOL NUEVO------------" << endl;
69      reflexion(arb,arb.root());
70      MostrarArbol(arb,arb.root());
71  }
```