```
1 #include "bintree.h"
 2 #include <iostream>
 3
4 #include <cmath>
 5 #include <queue>
6 #include <iomanip>
8 using namespace std;
9
10
11
12 template <class T>
13 void MostrarArbol(const bintree<T> &A, typename bintree<T>::node root)
14 {
15
       queue<typename bintree<T>::node> colaNodos;
16
       int totalNodos = A.size();
17
       int techo = log2(totalNodos + 1);
       colaNodos.push(root);
18
19
       int pot = 0;
20
       while (colaNodos.size() > 0)
21
           int niveles = colaNodos.size();
22
23
           while (niveles > 0)
24
25
               typename bintree<T>::node nodoAux = colaNodos.front();
               colaNodos.pop();
26
27
               cout << setw((niveles == pow(2, pot)) ? pow(2, (techo - pot)) : pow(2,
   (techo - pot + 1)));
28
               cout << *nodoAux;</pre>
29
               if (!nodoAux.left().null())
                    colaNodos.push(nodoAux.left());
30
31
               if (!nodoAux.right().null())
32
                    colaNodos.push(nodoAux.right());
33
               niveles--;
34
           }
35
           pot++;
           cout << endl;</pre>
36
37
       }
38 }
39
40 int profundidad(bintree<int>::node v)
41 {
42
       int prof=0;
43
       typename bintree<int>::node aux=v;
44
       while(!aux.parent().null()){
45
           prof++;
46
           aux=aux.parent();
47
48
       return prof;
49 }
50
51
52 bintree<int>::node ancestro_comun(bintree<int>::node n1, bintree<int>::node n2)
53 {
       static int prof_node1=profundidad(n1), prof_node2=profundidad(n2);
54
55
       if(prof_node1==-1 && prof_node2==-1)
56
           prof node1=profundidad(n1);
57
           prof_node2=profundidad(n2);
58
59
```

localhost:4649/?mode=clike 1/2

```
14/1/2020
                                                ejercicio15.cpp
  60
         if(prof_node1==prof_node2)
  61
             if(n1==n2)
  62
  63
             {
  64
                 prof_node1=-1; //RESET
  65
                 prof_node2=-1;
                 return n1; //DA IGUAL CUAL DEVOLVER PORQUE AMBOS SON IGUALES
  66
  67
  68
             {
  69
                 prof_node1--;
  70
                 prof_node2--;
  71
                 return ancestro_comun(n1.parent(),n2.parent());
  72
  73
         }else
  74
  75
             if(prof_node1 > prof_node2)
  76
             {
                 prof_node1--;
  77
  78
                 return ancestro_comun(n1.parent(),n2);
  79
             }else
  80
             {
  81
                 prof_node2--;
  82
                 return ancestro_comun(n1,n2.parent());
  83
             }
  84
         }
  85
  86 }
  87
  88 int main()
  89 {
  90
         bintree<int> arb2(0);
         arb2.insert_left(arb2.root(), 1);
  91
  92
         arb2.insert_right(arb2.root(), 2);
  93
  94
         bintree<int>::node aux2 = arb2.root().left();
  95
         arb2.insert left(aux2, 3);
  96
         arb2.insert_right(aux2, 4);
  97
         bintree<int>::node aux= aux2.right();
  98
 99
         aux2 = arb2.root().right();
 100
         arb2.insert left(aux2, 5);
 101
         arb2.insert_right(aux2, 6);
 102
         bintree<int>::node aux3= aux2;
 103
 104
 105
 106
         cout << "----" << endl;</pre>
         MostrarArbol(arb2,arb2.root());
 107
 108
         cout << "----" << endl;</pre>
 109
 110
         cout << *aux << *aux3 << endl;</pre>
 111
         cout << *(ancestro_comun(aux,aux3)) << endl;</pre>
 112
 113
```

localhost:4649/?mode=clike 2/2

114 }