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

#include <cmath>
#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 <typename T>
void inordenA_B(const bintree<T> &Arbol, typename bintree<T>::node v, T A, T B, queue<T> &dev)
{
    if (!v.null())
    {
        inordenA_B(Arbol, v.left(), A, B, dev);
        if (*v > A && *v < B)
        {
            dev.push(*v);
        }
        inordenA_B(Arbol, v.right(), A, B, dev);
    }
}

template <typename T>
queue<T> comprendidosA_B(const bintree<T> &Arbol, typename bintree<T>::node v, T A, T B)
{
    queue<T> dev;

    inordenA_B(Arbol, Arbol.root(), A, B, dev);
```

```
58
59      return dev;
60  }
61
62  int main()
63  {
64
65      bintree<int> arb(42);
66
67      bintree<int>::node aux;
68
69      //RAMA IZQUIERDA
70
71      arb.insert_left(arb.root(), 24);
72
73      aux = arb.root().left();
74
75      arb.insert_left(aux, 22);
76
77      arb.insert_right(aux, 35);
78
79      aux = aux.left();
80
81      arb.insert_left(aux, 3);
82
83      arb.insert_right(aux, 23);
84
85      aux = aux.parent().right();
86
87      arb.insert_left(aux, 33);
88
89      arb.insert_right(aux, 37);
90
91      //RAMA DERECHA
92
93      arb.insert_right(arb.root(), 57);
94
95      aux = arb.root().right();
96
97      arb.insert_left(aux, 51);
98
99      arb.insert_right(aux, 63);
100
101     aux = aux.left();
102
103     arb.insert_left(aux, 45);
104
105     arb.insert_right(aux, 55);
106
107     aux = aux.parent().right();
108
109     arb.insert_left(aux, 60);
110
111     arb.insert_right(aux, 72);
112
113     cout << "---------------ARBOL BST-----------------" << endl;
114
115     MostrarArbol(arb, arb.root());
116
117     int a, b;
```

```
118     a = 4;
119     b = 52;
120 
121     cout << "Comprendidos entre " << a << " y " << b << " -->" << endl;
122 
123     queue<int> otro = comprendidosA_B(arb, arb.root(), a, b);
124     while (!otro.empty())
125     {
126         cout << otro.front() << " ";
127         otro.pop();
128     }
129 }
```