```cpp
#include <iostream>
#include <string>
#include <list>

using namespace std;

template <typename T>
void mostrar_lista(const list<T> &l)
{
    typename list<T>::const_iterator p;
    for (p = l.begin(); p != l.end(); p++)
    {
        cout << *p << " ";
    }
    cout << endl;
}

template <typename T>
bool contenida(const list<T> &l1, const list<T> &l2)
{
    bool ret=false;
    typename list<T>::const_iterator p; //LISTA 2
    typename list<T>::const_iterator q=l1.cbegin(); //LISTA 1
    for(p=l2.cbegin();p!=l2.cend();p++) //BUSCAMOS EN QUE POSICION SE ENCUENTRA EL PRIMER ELEMENTO DE LA LISTA 1 EN LA LISTA 2
    {
        if(*p==*q)
        {
            ret=true;
            break;
        }
    }
    if(ret) //SI ESTA EN LA LISTA SEGUIMOS
    {
        while(p!=l2.cend() && q!=l1.cend() && ret) //BUCLE QUE ITERA TANTAS VECES COMO EL TAMAÑO DE LA LISTA 1 O HASTA QUE SE ACABE LA LISTA2 O SE PONGA A FALSE RET
        {
            p++;
            if(*p!=*q) //SI HA CAMBIADO DE VALOR INCREMENTAMOS Q (PARA CUANDO APARECEN POR EJEMPLO VARIOS 1 SEGUIDOS)
            {
                q++;
                if(q==l1.cend()) //SI HEMOS INCREMENTADO Y YA SE HA ACABADO NOS SALTAMOS LA COMRPOBACION PARA SALIR
                {
                    continue;
                }
                if(*p!=*q) //SI YA NO COINCIDEN PONEMOS RET A FALSE Y SALE FUERA
                {
                    ret=false;
                }
            }
            cout << *p << *q << endl;
        }
        if(p==l2.cend() && q!=l1.cend()) //SI HA SALIDO PORQUE SE HA ACABADO LA LISTA 2 PERO NO LA LISTA 1 PONEMOS RET A FALSE
        {
            ret=false;
        }
    }
```

```
56      return ret;
57  }
58
59  int main()
60  {
61      list<int> lista1;
62      lista1.push_back(1);
63      lista1.push_back(3);
64      lista1.push_back(8);
65      list<int> lista2;
66      lista2.push_back(2);
67      lista2.push_back(1);
68      lista2.push_back(1);
69      lista2.push_back(3);
70      lista2.push_back(3);
71      lista2.push_back(3);
72      lista2.push_back(8);
73      lista2.push_back(8);
74      lista2.push_back(1);
75      contenida(lista1,lista2)? cout << "contenida" << endl: cout << "no contenida"<<
    endl;
76  }
```