Grupo GR05

Estudiantes:

- EMBID SÁNCHEZ, JUAN (E17)
- VALENCIA BLANCAS, ALEJANDRO (E68)

ID envio	Usuario/a	Hora envío	Veredicto
52482	E68	2022-02-11T11:08:02.889	AC
52458	E68	2022-02-11T10:45:54.574	AC
52401	E68	2022-02-11T10:24:37.714	WA
52368	E68	2022-02-11T09:56:41.555	WA
52360	E68	2022-02-11T01:33:32.271	WA
52357	E68	2022-02-11T01:06:13.600	WA
52121	E68	2022-02-10T12:50:14.155	TLE
52118	E68	2022-02-10T12:39:47.193	TLE
52026	E68	2022-02-09T22:45:26.587	TLE
51529	E17	2022-02-07T16:16:01.207	CE
50678	E68	2022-02-02T16:31:46.410	RTE

Fichero `Source.cpp`

```
/*
 * ----------------------------------------------------
 *                 ESTRUCTURAS DE DATOS
 * ----------------------------------------------------
 *              Facultad de Informática
 *         Universidad Complutense de Madrid
 * ----------------------------------------------------
 */

 /*
  Introduce aquí los nombres de los componentes del grupo:

  Componente 1: Alejandro Valencia Blancas
  Componente 2: Juan Embid Sánchez
  */
//COSTE DEL PROBLEMA: O(n*log n)

#include <iostream>
#include <cassert>
#include <fstream>
#include <vector>

using namespace std;

const int MAX_ELEMS = 2000;


class Multiconjunto {
```

Se pide el coste de cada método del TAD

```
public:
    // No olvides el coste!

    bool pertenece(int elem) {
        int pos = buscar(elem, 0, num_elems);
        return elem == elems[pos].valor && elems[pos].multiplicidad > 0;
    }

    void anyadir(int elem) {
        if (pertenece(elem)) {
            elems[buscar(elem, 0, num_elems)].multiplicidad++; //si pertenece aumenta
    multiplicidad
        }
        else {
            if (num_elems == 0) { //si no pertenece y no hay elementos lo mete en la primera
    posición
                elems[0].valor = elem;
                elems[0].multiplicidad++;
                num_elems++;
            }
            else { //si no pertenece y si hay elementos
                for (int i = num_elems; i > buscar(elem, 0, num_elems) && buscar(elem, 0,
    num_elems) != num_elems; i--) {
                    elems[i] = elems[i - 1];
                }
                elems[buscar(elem, 0, num_elems)].valor = elem;
                elems[buscar(elem, 0, num_elems)].multiplicidad = 1;
                num_elems++;
            }
        }
    }

    void eliminar(int elem) {

        elems[buscar(elem, 0, num_elems - 1)].multiplicidad--;



    }

private:
    bool ok = false;

    int buscar(int elem, int inicio, int fin) {
        int centro = (fin + inicio) / 2;

        if (fin < inicio) {
            return centro;
        }

        if (elem < elems[centro].valor) {
            return buscar(elem, inicio, (centro - 1));
```

pos < num_elems &&

Llamad a buscar solo una vez al principio de la función.

se hace lo mismo aquí

¿Y si no está?

Además, si la multiplicidad se hace 0, hay que quitarlo del vector. Lo dice el invariante del enunciado.

```
        }

        if (elem > elems[centro].valor) {
            return buscar(elem, (centro + 1), fin);
        }

        if (elem == elems[centro].valor && elems[centro].multiplicidad > 0) {
            return centro;
        }

        return centro;

    }

    struct Elem {
        int valor;
        int multiplicidad = 0;
    };
    Elem elems[MAX_ELEMS];
    int num_elems = 0;
};

bool tratar_caso() {
    // Implementar
    Multiconjunto m;
    int aux[MAX_ELEMS], aux2[MAX_ELEMS]; //aux primer array sin ordenar, aux2 segundo array
    sin ordenar
    int num;
    cin >> num;
    if (num != 0) {
        for (int i = 0; i < num; i++) {
            cin >> aux[i];;
            m.anyadir(aux[i]);
        }

        for (int i = 0; i < num; i++) {
            cin >> aux2[i];
        }

         for (int i = 0; i < num; i++) {
            if (aux2[i] == aux[i])
                m.eliminar(aux[i]);
        }

        for (int i = 0; i < num; i++) {
            if (aux2[i] == aux[i]) {

                cout << "#";
            }
            else if (m.pertenece(aux2[i])) {
                m.eliminar(aux2[i]);
```

```
                cout << "O";
            }
            else {
                cout << ".";
            }
        }
        cout << "\n";
    }
    else  return false;

    return true;
}

int main() {
#ifndef DOMJUDGE
    std::ifstream in("sample.in");
    auto cinbuf = std::cin.rdbuf(in.rdbuf());
#endif
    while (tratar_caso()) {}

#ifndef DOMJUDGE
    std::cin.rdbuf(cinbuf);
    // Descomentar en Windows si la consola se cierra inmediatamente
    // system("PAUSE");
#endif
    return 0;
}
```