

# Encontrar un patrón en una secuencia

Se pide implementar una función semejante a la función `search` de la librería `algorithm`.

```
template < class T >
    T Mysearch(T const & ini1, T const & fin1, T const & ini2, T const & fin2);
```

Esta función busca en el rango `[ini1..fin1)` la primera aparición de la secuencia definida por `[ini2..fin2)` y devuelve un iterador al primer elemento de dicha aparición o `fin1` si la secuencia no se encuentra. La función debe ser genérica para cualquier tipo lineal que tenga definido un iterador.

La función recibe:

- `ini1`. Un iterador al principio de la secuencia en la que debemos buscar.
- `fin1`. Un iterador al final de la secuencia en la que se debe buscar.
- `ini2`. Un iterador al comienzo de la secuencia que queremos buscar.
- `fin2`. Un iterador al final de la secuencia que queremos buscar.

La función devuelve un iterador al comienzo de la primera aparición de la secuencia a buscar en la secuencia en que estamos buscando o `fin1` si no se encuentra la secuencia buscada.

Un ejemplo de llamada a esta función es:

```
std::list <int> lista1;std::list <int > lista2;
auto it = Mysearch(lista1.begin(), lista1.end(), lista2.begin(), lista2.end());
```

En este problema la función se utilizará para eliminar todas las apariciones de la segunda secuencia de entrada en la primera.

## Entrada

La entrada está formada por una serie de casos. Cada caso comienza con dos valores  $N1, N2$  que indican el número de valores de la primera y segunda secuencia que nos dan a continuación. En la línea siguiente nos dan los  $N1$  valores de la secuencia en que debemos buscar. En la línea siguiente nos dan los  $N2$  valores de la secuencia que buscamos. La entrada termina con dos ceros.

Se garantiza que  $0 < N1$  y  $0 < N2 < 10$ . Los valores de las secuencias pueden almacenarse en una variable de tipo `int`.

## Salida

Para cada caso se escribirá una línea con los valores que quedan en la primera secuencia después de eliminar todas las apariciones de la segunda secuencia.

## Entrada de ejemplo

```
10 2
3 5 7 9 3 5 4 6 3 5
3 5
6 3
1 2 3 1 2 3
1 2 3
6 1
1 2 3 1 2 3
3
5 3
1 2 3 4 5
4 5 6
0 0
```

### Salida de ejemplo

7	9	4	6	
1	2	1	2	
1	2	3	4	5

**Autor:** Isabel Pita