Arreglos

Son idénticos a los arreglos en c

declaración

tipo nombre[tamaño];
Ejemplo: int a[10];

posiciones

Las posiciones van del 0 al n-1
Si el arreglo tiene tamaño 3, accedes al primer, segundo y tercer elemento de la forma a[0], a[1], a[2] respectivamente

inicialización

Si puede inicializar en 0 si lo declaras globalmente

String (std::string)

Declaración

string nombre

Posiciones

Igual que en un arreglo

Inicialización

s = "el texto"

Editar

s = "probemos";
s[4] = 'a'; → Notar diferencia entre 'a' y "a"
s = "probamos";

String (std::string)

Concatenar

Sean s y t dos strings, la operación s + t concatena s con t
Ejemplo: sea s = "ab" y t = "bc", → s + t = "abbc"

Mayus y Minus

Sea s = "aB", ejecuto
s[0] = toupper(s[0]);
s[1] = tolower(s[1]);
ahora s = "Ab"

Tamaño

con s.length() averiguamos cuántos caracteres tiene el string s.
Ejemplo: s = "abcd" → s.length() = 4

C++ Sort

Sintaxis

- arreglos:
  sort(a,a+n);
- string:
  sort(s.begin(),s.end())

Orden

Por defecto se ordena de menor a mayor si son números, y orden lexicográfico si son strings.
Para ordenar de mayor a menor, se escribe:

- arreglos:
  sort(a,a+n,std::greater<>());
- string:
  sort(s.begin(),s.end(),std::greater<>())

Ejemplos

```
1   #include <iostream>
    #include <algorithm>
3   using namespace std;
    int main() {
5       int a[10] = {10,2,3,4,5,6,7,8,9,1};
        sort(a, a+10, std::greater<int>());
7       for(int i = 0; i < 10; i++)
            cout << a[i] << " ";
9       cout << endl;
    }
11
```

Ejemplos

```
#include <iostream>
#include <algorithm>
using namespace std;
int main() {
    int a[10] = {10,2,3,4,5,6,7,8,9,1};
    sort(a, a+10, std::greater<int>());
    for(int i = 0; i < 10; i++)
        cout << a[i] << " ";
    cout << endl;
}
```

Output: 10 9 8 7 6 5 4 3 2 1

Ejemplos

```
#include <iostream>
#include <algorithm>
using namespace std;

int main() {
    string s = "9513246780";
    sort(s.begin(), s.end());
    cout << s << endl;
}

```

Ejemplos

```
#include <iostream>
#include <algorithm>
using namespace std;

int main() {
    string s = "9513246780";
    sort(s.begin(), s.end());
    cout << s << endl;
}

```

Output: 0123456789

Ejercicio: Codeforces 141/A

Enunciado

Había un cartel con los nombres de dos personas. Se cayeron todas las letras del cartel al piso. No sabemos si alguien se llevó algunas letras, o si alguien pasó y tiró más letras abajo del cartel. Queremos saber si usando exactamente las letras encontradas debajo del cartel es posible restaurar ambos nombres (no tiene que faltar ninguna letra y no deben haber letras de más).

Input

Consiste en 3 líneas. La 1era tiene el nombre del primer invitado, la 2nda el nombre del segundo, y la 3era las letras que se encontraron en el piso. Ninguna línea está vacía y todas las letras son mayúsculas latinas. La longitud de cada línea no excede los 100 caracteres.

Ejercicio: Codeforces 141/A

Enunciado

Había un cartel con los nombres de dos personas. Se cayeron todas las letras del cartel al piso. No sabemos si alguien se llevó algunas letras, o si alguien pasó y tiró más letras abajo del cartel. Queremos saber si usando exactamente las letras encontradas debajo del cartel es posible restaurar ambos nombres (no tiene que faltar ninguna letra y no deben haber letras de más).

Output

Imprime "YES" si las letras pueden ser permutadas para obtener los nombres y "NO" en caso contrario.

Ejercicio: Codeforces 141/A

Primera Idea

`http://codeforces.com/contest/141/submission/35791066`

Solución

```cpp
#include <iostream>
#include <algorithm>
using namespace std;
int main() {
  string a,b,c;
  cin >> a >> b >> c;
  string d = a+b;
  sort(c.begin(), c.end());
  sort(d.begin(), d.end());
  if (c == d)
    (cout<<"YES"<<endl);
  else
    (cout<<"NO"<<endl);
}
```

Una estrategia clásica para ver si dos palabras son anagramas es ordenar ambas y comparar el resultado.

Arreglos multidimensionales

http://codeforces.com/problemset/problem/38/B

Dos piezas de ajedrez, una torre y un caballo, estan en un tablero de ajedrez. No se sabe donde estan, pero tenes garantizado de que ninguna de las dos puede comer a la otra. Se te pide encontrar la cantidad de formas de poner otro caballo en el tablero tal que ninguna pieza pueda comer a otra. Solo podes poner el caballo en un espacio vacío.

Input

La primera línea contiene la posición de la torre y la segunda la posición del caballo.

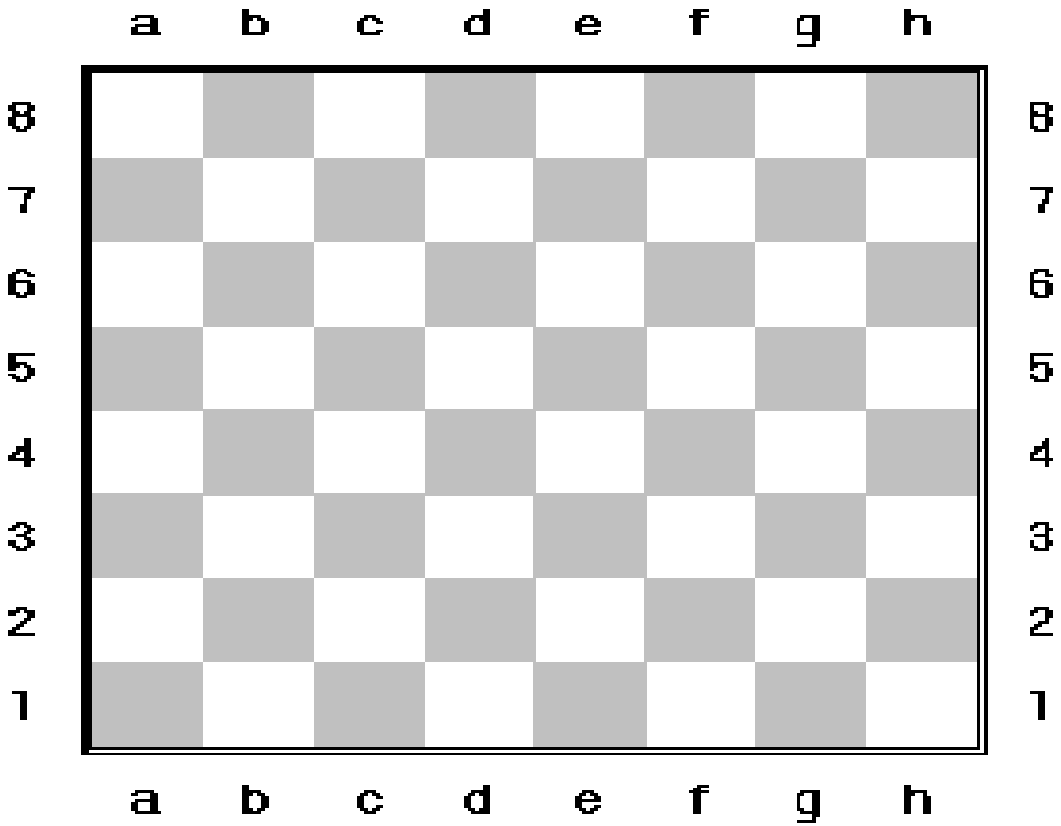
Output

Un único número, la cantidad de formas posibles.

Arreglos multidimensionales

¿Cómo representar el tablero?

Ver en pizarrón



Soluciones

Primera solución (implementación básica)

`http://codeforces.com/contest/38/submission/36119987`

Segunda solución (La primera solución mejor escrito)

`http://codeforces.com/contest/38/submission/36120296`

Tercera solución (Matriz implicita)

`http://codeforces.com/contest/38/submission/36336316`

Arreglos multidimensionales

Multiples dimensiones

Un arreglo de dos dimensiones se llama matriz
Podes hacer arreglos con n dimensiones
Por ejemplo: Un cubo rubix tiene 3 dimensiones y se escribe
int cubo[3][3][6]

Ejercicios de Arreglos, Strings, y Sort

`http://codeforces.com/problemset/problem/339/A`
`http://codeforces.com/problemset/problem/149/A`
`http://codeforces.com/problemset/problem/71/A`
`http://codeforces.com/problemset/problem/688/B`
`https://www.codechef.com/INSM2014/problems/INSM06`