



Aviso

Para todos los ejercicios:

- Los límites están disponibles en la página "Overview" de la plataforma del concurso.
- Hay un archivo adjunto que se puede descargar de la plataforma; contiene evaluadores de ejemplo, implementaciones de ejemplo, ejemplos de casos de prueba, y scripts de compilación y de ejecución.
- Puedes hacer hasta 50 envíos para cada ejercicio, y tienes que enviar exactamente un archivo cada vez.
- Cuando pruebes los programas con el evaluador de ejemplo, tu entrada debe coincidir con el formato y las restricciones del enunciado del ejercicio, de lo contrario, pueden producirse comportamientos no esperados.
- En las entradas del evaluador de ejemplo, dos tokens consecutivos en una línea están separados por un único espacio, a menos que se especifique otro formato explícitamente.
- Cuando pruebes tu código en tu ordenador, te recomendamos que utilice los scripts de los archivos adjuntos. Tenga en cuenta que utilizamos el parámetro `-std=gnu++17` al compilar.
- Si no puedes enviar tu código a CMS, puedes utilizar el comando `ioisubmit` para que se guarde y se evalúe una vez finalizado el concurso.
 - Ejecuta `ioisubmit <shortname_del_ejercicio> <archivo>` en el directorio que contiene `<archivo>`.
 - Pide a un miembro del comité que haga una foto de la salida de `ioisubmit`. Tu envío no será válido si no se ha realizado este paso.
 - Si estás compitiendo online, pide a tu supervisor que haga una foto de la salida de `ioisubmit` y la envíe a los organizadores.

Convención

Los enunciados de las tareas especifican cabeceras utilizando nombres de tipos de dato genéricos `void`, `bool`, `int`, `int[]` (arreglo), y `union(bool, int[])`.

En C++, los graders usan tipos de datos o implementaciones apropiadas, como se indica a continuación

<code>void</code>	<code>bool</code>	<code>int</code>	<code>int[]</code>
<code>void</code>	<code>bool</code>	<code>int</code>	<code>std::vector<int></code>

<code>union(bool, int[])</code>	length of array a
<code>std::variant<bool, std::vector<int>></code>	<code>a.size()</code>

En C++, `std::variant` se define en el encabezado `<variant>`. Una función de tipo `std::variant<bool, std::vector<int>>` puede devolver un `bool` o un `std::vector<int>`. El código mostrado a continuación muestra tres ejemplos funcionales de métodos que devuelven `std::variant`

```
std::variant<bool, std::vector<int>> foo(int N) {
    return N % 2 == 0;
}
std::variant<bool, std::vector<int>> goo(int N) {
    return std::vector<int>(N, 0);
}
std::variant<bool, std::vector<int>> hoo(int N) {
    if (N % 2 == 0) {
        return false;
    }
    return std::vector<int>(N, 0);
}
```