Avisos

Para todas las tareas:

- Los límites están disponibles en la página "Overview" del sistema del concurso.
- Hay un paquete adjunto que puedes descargar en el sistema del concurso, que contiene los evaluadores de ejemplo, implementaciones de ejemplo, casos de ejemplo, y scripts de compilación y ejecución.
- Puedes hacer hasta 50 envíos para cada tarea, y tienes que subir exactamente un archivo en cada envío.
- Cuando pruebes tus programas en el evaluador de ejemplo, tu entrada debe cumplir con el formato y las restricciones de la tarea, de lo contrario podrían ocurrir comportamientos no esperados.
- En la entrada del evaluador de ejemplo, cada dos valores consecutivos en una línea estarán separados por un espacio, a menos que se indique lo contrario.
- Cuando pruebes tu código localmente en tu computadora, te recomendamos que utilices los scripts de los paquetes adjuntos. Por favor observa que utilizamos la opción de compilación `-std=gnu++17`.
- Si no es posible subir tu código a CMS, puedes utilizar la herramienta `ioisubmit` para guardar tu código para que sea evaluado al final del concurso.
    - Ejecuta `ioisubmit <nombre_tarea> <archivo_origen>` en la carpeta con el `<archivo_origen>`.
    - Pide a un miembro del comité que tomen una fotografía de la salida de `ioisubmit`. Tu envío no será considerado hasta que este paso sea completado.
        - Si estas compitiendo online, pidele a tu aplicador que tome la fotografía de la salida de `ioisubmit` y envíala a los organizadores.

Convención

Las redacciones de cada tarea especifican valores usando los nombres genéricos de tipos `void`, `bool`, `int`, `int[]` (arreglo), y `union(bool, int[])`.

En C++, los evaluadores utilizan los tipos de datos o implementaciones listadas a continuación:

`void`	`bool`	`int`	`int[]`
`void`	`bool`	`int`	`std::vector<int>`

union(bool, int[])	tamaño del arreglo a
std::variant<bool, std::vector<int>>	a.size()

En C++, `std::variant` se define en la cabecera `<variant>`. Un metodo que retorna un tipo `std::variant<bool, std::vector<int>>` puede regresar ya sea un `bool` o un `std::vector<int>`. El codigo de ejemplo de abajo muestra tres ejemplos de funciones que retornan `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);
}
```