Notas

Para todas las tareas:

- Los límites están disponibles en la página de "Overview" en el sistema de la competencia.
- Existe un paquete adjunto que usted puede descargar del sistema de competencia, que contiene evaluadores de ejemplo, implementaciones de ejemplo, casos de prueba de ejemplo y scripts para compilar y ejecutar sus programas.
- Usted puede realizar hasta 50 envíos por cada tarea, y tiene que enviar exactamente un archivo en cada envío.
- Cuando pruebe sus programas con el evaluador de ejemplo, su entrada debe coincidir con el formato y las restricciones especificadas en el enunciado del problema, de otra manera, comportamientos inesperados pueden ocurrir.
- En las entradas para el evaluador de ejemplo, cada dos tokens consecutivos en una línea están separados por un espacio simple, a menos que otro formato sea especificado de forma explícita.
- Cuando usted pruebe su código en su máquina local, le recomendamos utilizar los scripts en los paquetes adjuntos. Por favor note que utilizamos la opción `-std=gnu++17` en el compilador.
- Si usted no puede enviar su solución al CMS, usted puede utilizar la herramienta `ioisubmit` para almacenar su código para futura evaluación al finalizar la competencia.
    - Ejecute `ioisubmit <nombre_corto_tarea> <archivo_fuente>` en el directorio con el `<archivo_fuente>`.
    - Solicite a un miembro del comité organizador que tome una foto de la salida de `ioisubmit`. Su envío no será considerado a menos que este paso sea realizado.
        - Si usted compite de forma online, solicite a su responsable que tome una foto de la salida de `ioisubmit` y la envíe a los organizadores.

Convención

Los enunciados de los problemas especifican las firmas utilizando tipos genéricos `void`, `bool`, `int`, `int[]` (arreglo), y `union(bool, int[])`.

En C++, los evaluadores utilizan los tipos de dato apropiados o implementaciones como se listan a continuación

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

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

En C++, `std::variant` es definido en el header `<variant>`. Un método con el tipo de retorno `std::variant<bool, std::vector<int>>` puede retornar un `bool` o un `std::vector<int>`. El código de ejemplo a continuación 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);
}
```