Importante

Para todos los problemas:

- Los límites están disponibles en la página "Overview" del sistema de la competencia
- Hay un paquete adjunto que puede descargar desde el sistema de la competencia. Este contiene los graders de ejemplo, implementaciones de ejemplo, casos de ejemplo y scripts de compilación y ejecución.
- Puede realizar hasta 50 envíos por cada problema y debe enviar exactamente un archivo en cada envío.
- Cuando pruebe sus programas con el grader de ejemplo, su entrada debe coincidir con el formato y restricciones del enunciado del problema; en caso contrario, pueden darse comportamientos imprevistos.
- En las entradas de los graders de ejemplo, cada par de tokens consecutivos de una misma línea deben estar separados por un espacio a menos que se haya especificado explícitamente el uso de otro formato.
- Cuando pruebe su código en su entorno local, recomendamos que use los scripts de los paquetes adjuntos. Por favor, note que usamos la opción de compilador `-std=gnu++17`.
- Si no le es posible realizar envíos al sistema de CMS, puede usar la herramienta `ioisubmit` para almacenar su código y que este sea evaluado luego del final de la competencia.
    - Ejecute `ioisubmit <nombre_corto_del_problema> <nombre_del_archivo>` en el directorio que contenga `<nombre_del_archivo>`.
    - Pida a un miembro del comité que tome foto a la salida de `ioisubmit`. Su envío no será considerado a menos que este paso haya sido realizado.
        - Si está compitiendo de manera online, pídale a su supervisor que tome foto a la salida de `ioisubmit` y que la envíe a los organizadores.

Convención

Los enunciados de los problemas especifican los tipos de dato usando nombres genéricos `void`, `bool`, `int`, `int[]` (arreglo), y `union(bool, int[])`.

En C++, los graders usan tipos de datos o implementaciones apropiadas, como se muestra 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` está definido en la librería `<variant>`. Un método con tipo de retorno `std::variant<bool, std::vector<int>>` puede devolver bien un `bool` o un `std::vector<int>`. El código a continuación muestra tres ejemplos de funciones que devuelven un `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);
}
```