Carátula Día 2

Para todos los problemas:

- Los límites están disponibles en la página "Overview" del sistema de evaluación.
- Hay un comprimido adjunto que puede descargar del sistema de evaluación, que contiene evaluadores locales, implementaciones de ejemplo, casos de prueba de ejemplo y scripts de compilación.
- Puede realizar hasta 50 envíos para cada problema y debe enviar exactamente un archivo en cada envío.
- Al probar sus programas con los evaluadores locales, su entrada debe coincidir con el formato y las restricciones del enunciado del problema, de lo contrario, pueden ocurrir comportamientos no especificados.
- En las entradas del evaluador de ejemplo, todo par de tokens consecutivos en una línea están separados por un solo espacio, a menos que se especifique explícitamente otro formato.
- Cuando pruebe su código en la computadora local, le recomendamos que utilice los scripts en el comprimido adjunto. Tenga en cuenta que usamos la opción del compilador std=gnu++17.
- Si no puede enviar al CMS, puede usar la herramienta ioisubmit para almacenar su código para su evaluación después del final del concurso.
    - Ejecute `ioisubmit <task_shortname> <source_file>` en el directorio con `<source_file>`.
    - Pídale a un miembro del comite que tome una foto de la salida de ioisubmit. Su envío no se considerará a menos que se haya realizado este paso.

Convenciones

Los enunciados especifican las signaturas utilizando nombres genéricos para los tipos: `void`, `bool`, `int`, `int[]` (arreglo), y `union(bool, int[])`.

En C++, los evaluadores utlizan tipos de datos o implementaciones apropiadas, que se mencionan 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` se define en el encabezado `<variant>`.

Una función de tipo `std::variant<bool, std::vector<int>>` puede retornar tanto un `bool` como un `std::vector<int>`.

El siguiente código muestra tres ejemplos válidos 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);
}
```