Notas

Para todos os problemas:

- Os limites estão disponíveis na página "Overview" do sistema de avaliação.
- Existe um ficheiro (arquivo) em anexo que podes transferir do sistema de avaliação que contém avaliadores exemplo, exemplos de implementações, exemplos de casos de testes e scripts de compilação e execução.
- Podes fazer no máximo 50 submissões em cada problema e tens de enviar exatamente um único ficheiro em cada submissão.
- Quando testares os teus programas com o avaliador exemplo, o teu input deve corresponder ao formato e às restrições do enunciado, pois, caso contrário, podem ocorrer comportamentos indefinidos.
- Nos exemplos de input, cada dois tokens consecutivos na mesma linha são separados por um único espaço, a menos que outro formato seja explicitamente especificado.
- Quando testares o teu código na tua máquina local, recomendamos que uses os scripts do arquivo que te foi dado. Por favor nota que usamos a opção de compilação `-std=gnu++17`.
- Se não conseguires submeter via CMS, podes usar a ferramenta `ioisubmit` para armazenar o teu código para ser avaliado no final da prova.
    - Executa `ioisubmit <task_shortname> <source_file>` no diretório com o `<source_file>`.
    - Pede a um membro do comité para tirar uma fotografia do output do `ioisubmit`. A tua submissão não será considerada se este passo não for feito.
        - Se estás a competir à distância, pede ao teu vigilante para tirar uma foto do output do `ioisubmit` e enviá-la aos organizadores.

Convenção

Os enunciados dos problemas especificam assinaturas usando nomes genéricos para os tipos de dados `void`, `bool`, `int`, `int[]` (array), e `union(bool, int[])`.

Em C++ os avaliadores usam os tipos de dados ou as implementações apropriadas, conforme listado a seguir

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

union(bool, int[])	tamanho do array a
`std::variant<bool, std::vector<int>>`	`a.size()`

Em C++ `std::variant` está definido no header `<variant>`. Um método que devolve o tipo `std::variant<bool, std::vector<int>>` pode devolver ou um `bool` ou um `std::vector<int>`. O código exemplo abaixo mostra três exemplos funcionais de funções que devolvem `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);
}
```