Примітка

Для всіх задач:

- Обмеження доступні на сторінці "Overview" у системі тестування.
- Є вкладення, яке ви можете завантажити з тестувальної системи, яке містить приклад градерів, приклад імплементації рішень, приклади з умови, а також скрипти для компіляції та запуску.
- Ви можете зробити не більше 50 спроб на кожне завдання, вам потрібно відправляти рівно один файл на кожну спробу.
- Коли тестуєте вашу програму з прикладом градера, формат ваших вхідних даних має збігатися з тим форматом, який описаний в умові. Інакше - можлива непередбачена поведінка програми.
- Градер зчитує всі послідовні числа (або слова) через пробіл, якщо інше не сказано в умові.
- Коли ви запускаєте код на своїй локальній машині, ми рекомендуємо використовувати скрипти, які доступні у вкладенні. Зверніть увагу, що ми використовуємо прапорець `-std=gnu++17`.
- Якщо ви не можете відправити рішення в CMS, ви можете використовувати `ioisubmit` для відправки вашого коду після завершення контеста.
  - Запустіть `ioisubmit <task_shortname> <source_file>` у теці з `<source_file>`.
  - Попросіть члена комітета зробити фотографію виводу `ioisubmit`. Ваша спроба не буде зарахована, якщо ви це не зробите.
    - Якщо ви берете участь онлайн, то попросіть вашого проктора зробити фотографію виводу `ioisubmit` і відправити її організаторам.

Домовленості

В умовах використовуються наступні назви `void`, `bool`, `int`, `int[]` (масив), а також `union(bool, int[])`.

У C++, градери використовують наступні типи даних:

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

union(bool, int[])	довжина масиву a
std::variant<bool, std::vector<int>>	a.size()

У C++, `std::variant` визначається в хедері `<variant>`. Метод, що повертає тип `std::variant<bool, std::vector<int>>` може повернути або `bool`, або `std::vector<int>`. Приклад коду нижче показує три можливих варіанти повернення `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);
}
```