# Notice

Для всех задач:

- Ограничения доступны на вкладке "Overview" в тестирующей системе.
- Вы можете скачать прикреплённый архив из проверяющей системы, архив содержит примеры грейдера, примеры реализации, примеры из условия и скрипты для компиляции и запуска.
- По каждой задаче вы можете сделать не более 50 посылок, в каждой из которых вы должны отправить на проверку ровно один файл.
- Когда вы тестируете ваши программы, используя пример грейдера, ваши входные данные должны соответствовать формату и ограничениям из условия задачи, в противном случае возможно неопределенное поведение.
- Во входных данных примера грейдера каждые два последовательных токена в строке разделены одиночным пробелом, кроме случаев, когда иной формат явно определен.
- При локальном тестировании рекомендуется использовать скрипты из архива. Обратите внимание, что используется опция компиляции `-std=gnu++17`.
- Если вы не можете сделать посылку, используя CMS, можно использовать утилиту `ioisubmit` для того, чтобы сохранить код для проверки после конца соревнования.
    - Запустите `ioisubmit <task_shortname> <source_file>` в директории, содержащей `<source_file>`.
    - Попросите проктора сфотографировать вывод `ioisubmit` и послать его организаторам. Ваша посылка не будет засчитана, если этот шаг не будет сделан.

# Convention

В условиях задач при задании сигнатур функций используются обозначения типов `void`, `bool`, `int`, `int[]` (массив) и `union(bool, int[])`, соответствие типам в C++ приведено в таблице:

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

| union(bool, int[]) | length of array a |
|---|---|
| `std::variant<bool, std::vector<int>>` | `a.size()` |

В языке std::variant определяется в заголовочном файле <variant>. Функция с типом возвращаемого значения std::variant<bool, std::vector<int>> может вернуть как bool, так и std::vector<int>. Код ниже иллюстрирует примеры использования 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);
}
```