Uwagi

Dla wszystkich zadań:

- Limity są dostępne na stronie "Overview" w systemie zawodów.
- Można ściągnąć dostępny w systemie zawodów pakiet zawierający przykładowe programy oceniające, przykładowe implementacje, przykładowe przypadki testowe i skrypty do kompilowania i uruchamiania kodów.
- Możesz wykonać co najwyżej 50 zgłoszeń każdego zadania. W każdym zgłoszeniu może być tylko jeden plik.
- Gdy używasz przykładowego programu oceniającego, musisz sam zadbać o poprawność danych. Jeśli dane są niepoprawne, działanie programu oceniającego jest nieokreślone.
- W przykładowym programie oceniającym kolejne elementy wejścia w każdym wierszu oddziela pojedyncza spacja, chyba że wprost powiedziano inaczej.
- Gdy testujesz Twój kod na lokalnym komputerze, zalecamy użycie skryptów z dostępnego pakietu. Zwróć uwagę na opcje kompilacji, których używamy: `-std=gnu++17`.
- Jeśli nie jesteś w stanie dokonać zgłoszenia do CMS, możesz użyć narzędzia `ioisubmit`, aby zapisać Twój kod do oceny po końcu zawodów.
  - Wywołaj wtedy `ioisubmit <task_shortname> <source_file>` w katalogu zawierającym `<source_file>`.
  - Poproś osobę z obsługi zawodów, aby wykonała zdjęcie wyniku wywołania `ioisubmit`. Jeśli pominiesz ten punkt, Twoje zgłosznie nie będzie rozważone.
    - Jeśli bierzesz udział zdalnie, poproś osobę pilnującą, aby wykonała zdjęcie wyjścia `ioisubmit` i wysłała organizatorom.

Konwencja

Opisy zadań zawierają sygnatury z użyciem generycznych nazw typów `void`, `int`, `int64`, `int[]` (tablica), oraz `union(bool, int[])`.

W języku C++, sprawdzaczki używają odpowiednich typów danych lub implementacji, tak jak w poniższej tablicy

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

union(bool, int[])	długość tablicy a
`std::variant<bool, std::vector<int>>`	`a.size()`

W C++, `std::variant` jest zdefiniowany w pliku nagłówkowym `<variant>`. Wynikiem działania funkcji, której wynikiem działania jest wartość typu `std::variant<bool, std::vector<int>>`, może być `bool` albo `std::vector<int>`. Poniższy kod przykładowy pokazuje trzy działające przykłady funkcji, których wyniki działania są typu `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);
}
```