Chú ý

Đối với tất cả các bài thi:

- Các giới hạn được thông báo ở trang "Overview" trên hệ thống thi.
- Có một gói đính kèm mà bạn có thể tải về từ hệ thống chứa các trình chấm mẫu, các cài đặt mẫu, các ví dụ kiểm thử, và các kịch bản biên dịch.
- Bạn có thể nộp tối đa 50 lần cho mỗi bài, và bạn phải nộp mỗi lần đúng một file.
- Khi thử nghiệm các chương trình của bạn với trình chấm mẫu, dữ liệu của bạn cần đáp ứng về khuôn dạng và ràng buộc như mô tả trong đầu bài, nếu không có thể xảy ra những kết quả bất thường.
- Trong dữ liệu vào của trình chấm mẫu, hai token liền kề nhau trên một dòng được phân biệt bởi một dấu cách, trừ khi một định dạng khác được mô tả cụ thể.
- Khi bạn thử nghiệm chương trình trên máy của bạn, chúng tôi khuyên bạn sử dụng các kịch bản dịch trong gói đính kèm. Lưu ý rằng chúng tôi sử dụng `-std=gnu++17` để dịch.
- Nếu bạn không thể nộp bài qua hệ thống CMS, bạn có thể sử dụng công cụ `ioisubmit` để lưu trữ bài làm để chấm sau khi kết thúc cuộc thi.
    - Thực thi lệnh `ioisubmit <mã_bài> <tệp_bài_làm>` trong cùng thư mục với `<tệp_bài_làm>`.
    - Yêu cầu giám thị chụp lại kết quả của lệnh `ioisubmit`. Bài nộp của bạn sẽ không được tính nếu thiếu bước này.
        - Nếu bạn thi online, yêu cầu giám thị chụp lại kết quả của lệnh `ioisubmit` và gửi cho ban tổ chức.

Quy ước

Đề bài mô tả quy cách các hàm dựa trên các kiểu dữ liệu chung `void`, `bool`, `int`, `int[]` (mảng), và `union(bool, int[])`.

Với ngôn ngữ C++, chương trình chấm sử dụng các kiểu dữ liệu phù hợp hoặc cài đặt như sau:

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

union(bool, int[])	**độ dài của mảng a**
`std::variant<bool, std::vector<int>>`	`a.size()`

Với ngôn ngữ C++, `std::variant` được định nghĩa bởi thư viện `<variant>`. Hàm với kiểu trả về là `std::variant<bool, std::vector<int>>` có thể trả về một biến kiểu `bool` hoặc một biến kiểu `std::vector<int>`. Đoạn mã nguồn ví dụ dưới đây gồm ba ví dụ về hàm trả về kiểu `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);
}
```