```
class blocking_flag {
public:
  blocking_flag() : ready_(false) {}

  void wait() {
    std::unique_lock<std::mutex> lock(mtx_);
    while (!ready_.load()) {
      ready_cond_.wait(lock);
    }
  }

  void set() {
    ready_.store(true);
    ready_cond_.notify_all();
  }

private:
  std::atomic<bool> ready_;
  std::mutex mtx_;
  std::condition_variable ready_cond_;
}
```

Нам нужно оценить корректность этого кода. Покажем что при некотором порядке исполнения инструкций код выполнится некорректно. Рассмотрим код из примера:

```
#include <thread>
#include <iostream>

int main() {
  blocking_flag f;

  std::thread t(
      [&f]() {
        f.wait();
        std::cout << "ready!" << std::endl;
      }
  );

  f.set();
  t.join();

  return 0;
}
```

И следующую последовательность инструкций:

1. Главный поток прерывается перед строчкой f.set()
2. Поток t вызывает f.wait(), захватывает mutex, проверяет условие while, заходит в while и прерывается
3. Главный поток вызывает f.set(), выполняет ready_.store(true) и ready_cond_.notify_all()
4. Поток t выполняет ready_cond_.wait(lock) и засыпает, так как notify_all() был вызван до wait()
5. Главный поток вызывает f.join()
6. Оба потока неограниченно долго находятся в системных вызовах ожидания, которые при нормальных условиях (то есть без spurious wakeup) не могут завершиться.