

Zadanie 3 (2pkt)

Co jest nie tak z poniższym kodem?

```
int Client::upload(const Item& i, Processing p, bool async)
{
    // sprawdzamy łącznienie
    if (isConnected() && isAuthorized() && (itemsQueue.size() == 0)
        && ((security == nullptr) || security->allowsUploads() || security->isAdmin()))
    {
        auto tmp = i.options[2].flags.value;

        if (tmp < 15) {
            tmp = 2 * i.size();
            switch (p) {
                case Processing::Async: return async ? startUpload(i, tmp) : -5;
                case Processing::Sync: return async ? -3 : uploadNow(i);
                default: return 0;
            }
        } else {
            switch (p) {
                case Processing::Async: return async ? startUpload(i, tmp - 2) : -5;
                case Processing::Sync: return async ? -3 : uploadNow(i);
                default: return 0;
            }
        }
    }
    else
    {
        if (parent.mainProvider.registry.items.cache.contains(i))
            return 2;
        return -4;
    }
    return -5;
}
```

Proszę opisać jak należałoby poprawić ten kod (jakie operacje i w jaki sposób wykonane, kod nie jest konieczny).

Uwagi do prowadzącego (K. Grochowskiego):

Zadanie 5 - współbieżne wzorce projektowe (2pkt)

Jakie zjawisko występuje w poniższym kodzie (wyścig/zakleszczenie/zagłodzenie/żadno z powyższych)?

Popraw kod w ramce. Kodu poza ramką proszę nie poprawiać, nie jest istotny.

```
using Data = int;
const Data EMPTY = -1; //reprezentuje brak danych
int main () {
    Process p1(100), p2(200), p3(300);
    p1.setPrev(&p3);
    p2.setPrev(&p1);
    p3.setPrev(&p2);
    thread thrd1( ref(p1) );
    thread thrd2( ref(p2) );
    thread thrd3( ref(p3) );
    thrd1.join();
    thrd2.join();
    thrd3.join();
    return 0;
}
```

```
class Process {
public:
    static const int NUM = 100; //liczba operacji

    Process(int id) : id_(id) {
        data_.push_back(id_);
        data_.push_back(id_);
    }
    void setPrev(Process* p) { prev_ = p; }

    Data get() {
        Data ret = EMPTY;
        lock_guard<mutex> lock(m_);
        if(!data_.empty() ) {
            ret = data_.front();
            data_.pop_front();
        }
        return ret;
    }
    void operator()() {
        if(!prev_) return; //nie ma zrodla danych

        for(int i=0;i<NUM;++i) {
            lock_guard<mutex> lock(m_);
            Data d = prev_->get();
            if( d != EMPTY ) {
                data_.push_back(d+1);
            }
        }
    }
private:
    int id_;
    Process* prev_ = nullptr;
    list<Data> data_;
    mutex m_;
};
```