Imię i nazwisko: | | | | | | | | | | | | | | | | | | | | | | | | | | **26L kol 2**

**Proszę wpisywać odpowiedzi w miejscach na to przeznaczonych**

**Zadanie 1 (2pkt)**

Proszę odpowiedzieć na poniższe pytania:

- W jaki sposób badać jakość testu?

- Czy (i ewentualnie jak) można zmierzyć "utrzymywalność" (maintainability) kodu?

**Zadanie 2 (2pkt)**

Dana jest funkcja:

```
void Cache::update(const Request& r, const std::size_t ttl) {
  if (r.empty() || (ttl == 0u))
    throw InvalidArgument{};

  if (r.timestamp() <= newestDataTs())
    return;

  if (hasItem(r))
    invalidate(r);

  cache(r, ttl);

  for (const auto& sub : r.subrequests())
    cache(sub, ttl);
}
```

Ile przypadków testowych należy napisać, by uzyskać:

- 100% pokrycia linii kodu,
- 100% pokrycia decyzji w kodzie.

Podaj elementy klas równoważności argumentów konieczne do uzyskania wspomnianych pokryć.

**Zadanie 3 (2pkt)**

Co jest nie tak z poniższym kodem?

```
class User {
public:
  std::string name;
  int age;

  int getYearOfBirth(int currentYear) {
    return currentYear - age;
  }
};


int Database::process(User* u) {
  if (u != nullptr) {
    if (u->age > 0) {
      auto x = u->getYearOfBirth(2026);
      if (x > 1900) {
        auto name = formatName(u->name);
        saveToDb(u);
        return log(name);
      } else {
        saveToDb(u);
        auto r = log(u->name);
        if (r != 0)
          return r;
        return 0;
      }
    }
    else
    {
      return -2;
    }
  } else {
    return -1;
  }
  return 0;
}
```

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

---

**Uwagi do prowadzącego (K. Grochowskiego):**

Imię i nazwisko: | | | | | | | | | | | | | | | | | | | | | | | | | | **26L kol 2**

**Proszę wpisywać odpowiedzi w miejscach na to przeznaczonych**

**Zadanie 4 - współbieżne wzorce projektowe (2pkt)**

Pokazana niżej klasa `Reader` ma metodę `read_data`, która czyta paczkę danych z dwóch portów, dane mogą przyjść na dowony z nich lub na oba. Kod kompiluje się poprawnie. Gdy dane nie przyjdą (koniec transmisji przez timeout) to klasa działa poprawnie. Gdy dane przychodzą, to pętla zdarzeń się nie kończy. Popraw kod w ramce.

```
using Time = asio::steady_timer::duration;
using Error = const error_code&;
using Bytes = std::vector<std::byte>;
class Reader {
public:
    Reader(const string& nameA, const string& nameB)
        : io_(), portA_(io_, nameA), portB_(io_, nameB), timer_(io_) {
    }
    ~Reader() { portA_.close(); portB_.close(); }

    const Bytes& read_data(size_t size, Time timeout) {
        timer_.expires_from_now(timeout);
        timer_.async_wait ( [=](Error e){this->timerEvent(e);} );
        portA_.async_read_some( asio::buffer(tabA, BUFFER_SIZE),
                            [=](Error e,size_t expected_size){ this->readA(e,size);});
        portB_.async_read_some( asio::buffer(tabB, BUFFER_SIZE),
                            [=](Error e,size_t expected_size){this->readB(e,size);});
        io_.run();
        return buffer_;
    }
private:
    asio::io_service io_;
    asio::serial_port portA_;
    asio::serial_port portB_;
    asio::steady_timer timer_;

    static const size_t BUFFER_SIZE = 1;
    std::byte tabA [BUFFER_SIZE];
    std::byte tabB [BUFFER_SIZE];
    Bytes buffer_;

    void readA(Error error, size_t expected_size) {
      if( error ) return;
      buffer_.push_back(tabA[0]); //wielkosc bufora = 1
      if(buffer_.size()>=expected_size){
        timer_.cancel();
        return;
      }
      portA_.async_read_some( asio::buffer(tabA, BUFFER_SIZE),
                           [=](Error e,size_t expected_size){ this->readA(e,expected_size);});
    }

    void readB(Error error, size_t expected_size) {
      if( error ) return;
      buffer_.push_back(tabB[0]);
      if(buffer_.size()>=expected_size){
        timer_.cancel();
        return;
      }
      portB_.async_read_some( asio::buffer(tabB, BUFFER_SIZE),
                            [=](Error e,size_t expected_size){this->readB(e,expected_size);});
    }

    void timerEvent(Error error) {
      if( error ) return;
      portA_.cancel();
      portB_.cancel();
    }
  };
```

**Zadanie 5 - współbieżne wzorce projektowe (2pkt)**

Popraw kod funkcji `thrd_main`. W systemie wiele wątków pobiera i przetwarza dane z obiektu typu `Vector`, zaś dane są wstawiane przez system zewnętrzny (jeden), który zużywa pomijalnie mało zasobów. Kodu systemu wstawiającego nie zamieszczono. Na platformie wieloprocesorowej ilość danych w kolejce jest duża i rośnie. Zwiększanie ilości wątków przetwarzających niewiele poprawia. Funkcja `calc` jest długa, zajmuje ponad 500ms dla paczki danych.

```
void calc(const Data& d); //dlugotrwale obliczenia

int main() {
  Vector v;
  //pominieto tworzenie watkow, ktore wstawiaja obiekty PData do kolejki
  const int NUM = 10;
  std::vector<std::thread> thrds;
  thrds.reserve(NUM);
  for (int i=0; i < NUM; ++i) {
    thrds.emplace_back(&thrd_main, ref(v));
  }
  //join_all
  for(int i=0; i< NUM; ++i)
    thrds[i].join();

  //... inny kod (nieistotny dla zadania)
}
```

```
class Vector {
public:
    bool empty_safe() {
        lock_guard lock(m_);
        return empty_impl();
    }
    bool empty_impl() { return v_.empty(); }
    PData read_safe() {
        lock_guard lock(m_);
        return read_impl();
    };
    PData read_impl() {
        PData out(nullptr);
        if(! empty_impl() ) {
            out = v_.back();
            v_.pop_back();
        }
        return out;
    }
    mutex m_;
    //nie zamieszczono metod do wstawiania elementow
private:
    std::vector<PData> v_;
};

void thrd_main(Vector& v) {
    while(! v.empty_safe()) {
        lock_guard l(v.m_);
        calc(*v.read_impl());
    }
}
```

**Uwagi do prowadzącego (R. Nowaka):**