Imię i nazwisko: | | | | | | | | | | | | | | | | | | | | | | | | | | **24L-2** 60 min

Przyjąć, że udostępniona jest przestrzeń nazw std i boost

W rozwiązaniach można używać dowolnych udogodnień z biblioteki standardowej C++ oraz bibliotek Boost.

Zadanie 1 (2pkt)

Obiekt typu `Queue` będzie implementowany albo jako `Queue1`, albo jako `Queue2` w następującym kodzie

```
Queue q;
Reader r1(q), r2(q), r3(q);
Writer w1(q), w2(q);
thread thrd1(ref(r1)); thread thrd2(ref(r2)); thread thrd3(ref(r3)); thread thrd4(ref(w1)); thread thrd5(ref(w2));
std::this_thread::sleep_for( chrono::milliseconds(1000));
r1.finish(); r2.finish(); r3.finish(); w1.finish(); w2.finish();
thrd1.join(); thrd2.join(); thrd3.join(); thrd4.join(); thrd5.join();
```

```
using Data = int; //Data to paczka danych do przetwarzania, wielkość paczki to ok. 1MB.
static const Data EMPTY = -1; //reprezentuje brak danych
using PNode = shared_ptr<Node>;
struct Node { Node(Data v, PNode n) : value(v), next(n) {} Data value; PNode next; };
struct Reader {
 Reader(Queue& q) : finish_(false), queue_(q) {}
 void finish() { finish_ = true; }
 void operator()() {
  while(!finish_) {
   Data d = queue_.read();
   //ten komentarz oznacza algorytm, który wykonuje się lokalnie
  }
 }
 Queue& queue_; volatile bool finish_;
};
struct Writer {
 Writer(Queue& q) : finish_(false), queue_(q) {}
 void finish() { finish_ = true; }
 void operator()() {
  while(!finish_) {
    //ten komentarz oznacza algorytm, który wykonuje się lokalnie, na koniec inicjuje obiekt d
    Data d;
    queue_.write(d);
  }
 }
 Queue& queue_; volatile bool finish_;
};
```

```
struct Queue1 {
 PNode head_; mutex mutex_;
 Data read() {
  Data out = EMPTY;
  lock_guard<mutex> lock(mutex_);
  if( head_ ) {
   out=head_->value; head_=head_->next;
  }
  return out;
 }
 void write(Data val) {
  lock_guard<mutex> lock(mutex_);
  PNode n = make_shared<Node>(val,head_);
  head_ = n;
 }
};
```

```
struct Queue2 {
 std::atomic<PNode> head_;
 Data read() {
  Data out = EMPTY;
  PNode node = head_.load();
  while(node && !head_.compare_exchange_weak(node, node->next))
       {}
  if( node ) out = node->value;
  return out;
 }
 void write(Data val) {
  PNode node = make_shared<Node>( val, head_.load() );
  while (!head_.compare_exchange_weak(node->next, node))
   {}
 }
};
```

- Czy `Queue1` jest wolne od wyścigów? [ ] Jeżeli nie, to w której funkcji (read czy write) i w której linii (read ma 6 linii, write ma 3) może być wyścig? [ ]
- Czy `Queue2` jest wolne od wyścigów? [ ] Jeżeli nie, to czy `Queue2` jest poprawne? [ ] Jeżeli znowu jest odpowiedź nie, to w której funkcji (read czy write) i w której linii (read ma 6 linii, write ma 3) jest problem? [ ]
- Czy `Queue2` jest bardziej wydajne, niż `Queue1`?

  ☐ TAK

  ☐ TAK, jeżeli (proszę dopisać) [ ]

  ☐ NIE

Zadanie 2 (1pkt)

Liczba liter Twojego nazwiska to stała NAME_SIZE.

`const int NAME_SIZE =` [ ]; Podaj wartość zwracaną przez funkcję `count(v)` [ ] dla wektora `vec`:

```
vector<V> vec;
vec.push_back(NAME_SIZE); vec.push_back(string("ZPR")); vec.push_back(1.0);

using V = variant<int,string,double>;
int count(const std::vector<V>& vec) {
   struct Visitor
   {
      Visitor() : counter_(0) {}
      void operator()(const int& i) { counter_ += i; }
      void operator()(const string& s) { counter_ += s.size(); }
      void operator()(const double& d) { }
      int counter_;
   } visitor;
   for_each(vec.begin(), vec.end(), [&](const V& v){ std::visit(visitor, v); } );
   return visitor.counter_;
};
```

Zadanie 3 (2pkt)

Popraw funkcję `read()`. Dostarczony kod kompiluje się i działa zgodnie z oczekiwaniami, za wyjątkiem konwersji liczby -1 i -2. Strumień jest skonfigurowany aby nie zgłaszać wyjątków, wykorzystujemy `operator>>(int& value)`, która dokonuje poprawnej konwersji, czyli pomija białe znaki, wczytuje cyfry i dostarcza liczbę całkowitą. Ustawia także flagi, które można badać, np. `failbit()` jest ustawiany, gdy błąd formatowania, np. litery zamiast cyfr.

Popraw funkcję, aby były poprawne konwersje dla wszystkich liczb ujemnych. Do sygnalizacji błędu wykorzystaj obiekty klasy `Error` (zamiast `int`), inicjowane odpowiednio na:

```
Error("Empty string");
Error("Bad string");
```

Niepoprawne napisy zdarzają się czasami, wykorzystanie wyjątków jest nie jest optymalnym rozwiązaniem, ale będzie uznane, jeżeli nie znajdziesz innego sposobu.

```
class Error : public std::exception {
public:
   Error(const string& m) : msg_(m) {}
   const std::string get() const { return msg_; }
   const std::string msg_;
};
```

```
static const int ERROR_EMPTY = -1;
static const int ERROR_BAD = -2;
int read(istream& is) {
   if(is.peek() == EOF) {
      return ERROR_EMPTY;
   }
   int out = 0;
   is >> out;
   if(is.fail() || is.bad() )
      return ERROR_BAD;
   else
      return out;
}
```

Pytanie 1 (1pkt)

Jakie treści proponujesz dodać do ZPR w przyszłych edycjach?

Ile godzin poświęciłeś na ZPR (uwzględnij wszystko, tzn. wykład, kolokwia, projekt i inne)? [ ]

Uwagi do prowadzącego (R.Nowak):

Imię i nazwisko: | | | | | | | | | | | | | | | | | | | | | | | | | | **24L-2** 60 min

Zadanie 4 (2pkt)

Zaproponuj strukturę danych i funkcję o jednym parametrze, która pozwala na skonfigurowanie połączenia z bazą danych [host, port, login, hasło (może być puste), nazwa bazy]. Użytkownik może pominąć dowolne pola z powyższego zestawu i w takim przypadku konfiguracja zostanie uzupełniona o wartości domyślne, które są odczytywane z pliku konfiguracyjnego. Jeżeli po scaleniu konfiguracja dalej jest niekompletna, funkcja ma zakończyć się błędem. W przeciwnym razie zwraca kompletną konfigurację.

Funkcja `get_connection_defaults` odczytuje domyślne wartości z pliku i zwraca je w postaci Twojej struktury danych. Załóż istnienie poniższej funkcji i wykorzystaj ją.

```
auto get_connection_defaults();
```

Zadanie 5 (2pkt)

Wymień cztery zalety używania algorytmów z biblioteki standardowej

Uwagi do prowadzącego (W. Wysota):