Imię i nazwisko: ..... **G** czas: 60 min

Przyjąć, że udostępniona jest przestrzeń nazw std

Zadanie 1 (2pkt)

Podany kod działa niepoprawnie dla pewnych danych. Zmień go tak, żeby działał zawsze. Zastosuj reguły borrow checker'a znane z Rust'a i/lub metody synchronizacji.

```
double median(std::vector<double> &values) {
   auto middle = values.begin() + std::ceil(values.size() / 2);

   // nth_element(first, nth, last) is a partial sorting algorithm that
   // rearranges elements in [first, last) such that:
   //
   // * The element pointed at by nth is changed to whatever element would
   // occur in that position if [first, last) were sorted.
   //
   // * All of the elements before this new nth element are less than or equal
   // to the elements after the new nth element.
   std::nth_element(values.begin(), middle, values.end());
   auto size = values.size();
   if ((size < 2) || (size % 2 == 1)) {
      return *middle;
   } else {
      return (*middle + *(middle - 1)) / 2;
   }
}

double mean(const std::vector<double> &values) {
   return std::accumulate(values.begin(), values.end(), 0.) / values.size();
}

std::pair<double, double> mean_and_median(std::vector<double> &values) {
   double x, y;

   std::thread t1([&x, &values]() { x = mean(values); });
   std::thread t2([&y, &values]() { y = median(values); });
   t2.join();
   t1.join();

   return std::make_pair(x, y);
}

int main() {
   std::vector<double> values = {};
   for (auto i = 1e5; i >= 0; --i) {
      values.push_back(i);
   }
   std::cout << mean(values) << std::endl;
   auto mean_median = mean_and_median(values);
   std::cout << mean_median.first << "␣" << mean_median.second << std::endl;
   return 0;
}
```

Zadanie 2 (2pkt)

Czym jest niezdefiniowane zachowanie? W jaki sposób można je wykryć w programie?

Uwagi do prowadzącego (Ł. Neumann):

Imię i nazwisko:..... **G** czas: 60 min

Przyjąć, że udostępniona jest przestrzeń nazw std

Zadanie 3 (2pkt)

Lista liczb całkowitych (klasa List i Node) działa niepoprawnie, przykład przedstawiono obok. Popraw kod listy i/lub węzła.

```
List l;
l.push_front(1);
l.push_front(2);
l.push_front(3);
```

```
class List {
   struct Node;
   using PNode = std::shared_ptr<Node>;
   using PWNode = std::weak_ptr<Node>;
   struct Node {
      Node(int v) : value_(v) {}
      ~Node() {}
      int value_;
      PWNode me_; //wskaznik na ten sam wezel
      PWNode next_; //element nastepny
   };
public:
   List() {}
   ~List() {}
   void push_front(int value) {
      PNode node( new Node(value) );
      node->me_ = node;
      if( head_ ) {
         node->next_ = head_;
         tail_->next_ = node;
      } else {
         node->next_ = node;
         tail_ = node;
      }
      head_ = node;
   }
private:
   PNode head_, tail_;
};
```

Zadanie 4 (1pkt)

Zainicjuj obiekt liczbą liter swojego nazwiska.

```
const int LICZBA_LITER_NAZWISKA =
```

Podaj wydruk programu przedstawionego obok.

```
class E : public std::exception {
public:
   E(int i) : i_(i) {}
   int i_;
};
struct B {
   B(int i) : i_(i) { cout << i_; }
   virtual ~B() { cout << i_; }
   int i_;
};
using PB = unique_ptr<B>;
B f(B& b) {
   return B(b.i_+1);
}
B f(B&& b) {
   throw E(b.i_);
}
void g(int i) {
   PB pb(new B(i));
   B b2 = f(*pb);
   B b3 = f(f(*pb));
}
int main() {
   try {
      g(LICZBA_LITER_NAZWISKA);
   } catch(E& e) {
      cout << e.i_;
   }
   cout << endl;
   return 0;
}
```

Uwagi do prowadzącego (R.Nowak):

Zadanie 5 (3pkt)

Dostarczamy 3 klasy do przetwarzania napisów. Popraw implementację aby wyeliminować powtórzenia. Pamiętaj o wzorcu NVI.

```
// klasa bazowa do konwersji napisow
class StrConverter {
public:
  StrConverter(const string& in) : INPUT(in) { }
  virtual std::string convert() {
    return doConvert(INPUT);
  }
private:
  virtual string doConvert(const string& in) = 0;
  const string& INPUT;
};

// zamienia male litery ASCII na ąodpowiadajce im wielkie litery
class StrToUpper : public StrConverter {
public:
  StrToUpper(const std::string& in) : StrConverter(in) {}
private:
  virtual string doConvert(const string& in) {
    string out;
    for(int i=0; i< in.size(); ++i) {
      out.push_back( toupper( in[i] ) ); //toupper zmienia na wielka litere. 'toupper' wystarczy w tym zadaniu
    }
    return out;
  }
};

// zamienia sekwencje bialych znakow na jedna spacje
class StrCutWhite : public StrConverter {
public:
  StrCutWhite(const std::string& in) : StrConverter(in) {}
private:
  virtual string doConvert(const string& in) {
    string out;
    bool drop = false;
    for( char c : in ) {
      if( isspace(c) ) { //w tym zadaniu 'isspace' wystarczy
        if( !drop ) {
          drop = true; //bedzie pomijal biale znaki
          out.push_back('␣');
        }
      } else { //to nie bialy znak
        drop = false;
        out.push_back(c);
      }
    }
    return out;
  }
};

//po kazdych N znakach wstawia znak nowej linii
class StrMaxLine : public StrConverter {
public:
  StrMaxLine(const string& in, unsigned int maxline) : StrConverter(in), MAX_LINE_(maxline) {}
private:
  const unsigned int MAX_LINE_; //przechowuje N
  virtual string doConvert(const string& in) {
    string out;
    unsigned int count = 0;
    for( string::const_iterator i = in.begin(); i < in.end(); ++i ) {
      out.push_back(*i);
      ++count;
      if(count >= MAX_LINE_) {
        out.push_back('\n');
        count = 0;
      }
    }
    return out;
  }
};
```