Imię i nazwisko:..... **W2** czas: 60 min

Przyjąć, że udostępniona jest przestrzeń nazw std, std::placeholders i boost

Zadanie 1 (5pkt)

W systemie obserwujemy niestabilne zachowanie aplikacji. Proszę usunąć problem. Przykład użycia obiektów `Thread` obok.

```
void zad1() {
  using Vs = vector<string>;
  Thread t1(Vs({"101","102","xxx"}));
  Thread t2(Vs({"201","20x","202"}));
  try {
    boost::thread thrd1( ref(t1) );
    boost::thread thrd2( ref(t2) );
    thrd1.join();
    thrd2.join();
  } catch(exception&) {
     cout << "error" << endl;
  }
}
```

```
class Thread {
public:
  Thread(const vector<string>& tasks) : tasks_(tasks) {}

  void operator()() {
    for( string s : tasks_ ) {
      int i = boost::lexical_cast<int>(s);
      //tutaj obliczenia
      boost::this_thread::sleep_for( boost::chrono::milliseconds(200));
    }
  }
private:
  vector<string> tasks_;
};
```

Zadanie 2 (6pkt)

Szablon `Vector` przechowuje elementy w specjalnym obszarze pamięci, wskazywanym przez składową `buffer`. Popraw implementację szablonu, aby pominąć pętlę w `~Vector`, gdy przechowujemy elementy, dla których wołanie destruktora może być pomijane. Fragment dokumentacji dla `is_trivially_destructible` i `boost::has_trivial_destructor` dodano poniżej. Obie klasy mają identyczne zastosowanie.

```
template < class T > struct std::is_trivially_destructible; (since C++11)
template < class T > struct boost::has_trivial_destructor;
```

If a type has a trivial destructor then the destructor has no effect: calls to the destructor can be safely omitted. `has_trivial_destructor<T>` derives from `true_type`, `has_trivial_destructor<T>::value` provides member equal to `true`, otherwise `has_trivial_destructor<T>` derives from `false_type`, `has_trivial_destructor<T>::value` equal to `false`.

```
template<typename T> class Vector {
public:
   typedef unsigned char BUF_ELEMENT;
   static const int BUFFER_SIZE = 100;

   Vector() : buffer_(new BUF_ELEMENT[BUFFER_SIZE]), size_(0) { }
   ~Vector() {
      for(int i=0;i<size_;++i) {
         T* t = reinterpret_cast<T*>(buffer_ + i*sizeof(T) );
         t->~T(); //wola destruktor
      }
      delete [] buffer_;
   }
   void add(const T& t) {
      new (buffer_ + size_*sizeof(T) ) T(t);
      ++size_;
   }
private:
   BUF_ELEMENT* buffer_;
   int size_;
};
```

Pytanie 1 (1pkt)

Wiele równoległych wątków jednocześnie czyta z tego samego kontenera typu `std::map`. Jaki wpływ na poprawność i skalowalność będzie miało przeprowadzanie tej operacji w sekcji krytycznej?

Zadanie 3 (3pkt)

NAME to Twoje nazwisko zapisane wielkimi literami ASCII (zamiast 'Ą' jest 'A'), np. `WROZKA` dla Wróżka.

```
const string NAME = "␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣;
```

Podaj napis, który zostanie wydrukowany przez funkcję `zad3`

```
typedef boost::variant<double,string> V;
int count(const std::vector<V>& v) {
   class Vis : public boost::static_visitor<void> {
   public:
      Vis() : sum_(0) {}
      void operator()(const double& d){ sum_ += static_cast<int>(d); }
      void operator()(const string& s) {
         try {
            sum_ += boost::lexical_cast<int>(s);
         } catch(boost::bad_lexical_cast &) {
         }
      }
      int sum_;
   } vis;
   for_each(v.begin(), v.end(), [&](const V& v){apply_visitor(vis, v);});
   return vis.sum_;
}
void zad3() {
   std::vector<V> v;
   v.push_back(1); v.push_back("100"); v.push_back(NAME); v.push_back(NAME + "1"); v.push_back(NAME.size());
   cout << count(v) << flush;
}
```

Zadanie 4 (3pkt)

Podaj napis, który zostanie wydrukowany przez funkcję `zad4`. Stałą `NAME` zdefiniowano w poprzednim zadaniu.

```
void zad4() {
   list<int> l;
   l.push_back(0); l.push_back(NAME.size()/7); l.push_back(NAME.size()); l.push_back(0);

   transform(l.begin(), l.end(), l.begin(), [](int i){ return i + 1; });
   l.erase( remove(l.begin(), l.end(), 1 ), l.end());
   copy(l.begin(), l.end(), ostream_iterator<int>(cout, ",") );
}
```

Zadanie 5 (5pkt)

Dostarcz funkcję `most_common`, która zwraca najczęściej występującą liczbę w wektorze liczb całkowitych. Jeżeli kolekcja jest pusta zwracamy 0. Użyj algorytmów z biblioteki standardowej.

```
int most_common(const std::vector<int>& numbers) {
```

Pytanie 2 (1pkt)

Zaproponuj zagadnienie, które Twoim zdaniem warto byłoby omówić na przedmiocie ZPR

Pytanie 3 (1pkt)

Ile godzin w semestrze poświęciłeś na przedmiot ZPR (wykład, projekt, kolokwia, nauka własna i inne)

Notatki / uwagi do prowadzącego