Imię i nazwisko: | | | | | | | | | | | | | | | | | | | | | | | | | | **23Z-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)

Zdefiniowano i zainicjowano liczniki pokazane niżej. Proszę podać minimalną wartość liczników, po wykonaniu kodu pokazanego w kolumnie 1. Jeżeli nie może być wyścigu, to wartość minimalna jest taka sama jak maksymalna. Kod z kolumny 1 jest wykonywany zawsze po inicjacji, gdy liczniki mają wartość 0.

```
int raw_c = 0;
int raw_c2 = 0;
atomic<int> atomic_c = 0;
atomic<int> atomic_c2 = 0;
mutex m;
```

```
void serve( int i) {
   raw_c += i;
   raw_c2 += raw_c;
}
void serve_m( int i) {
   lock_guard l(m);
   serve(i);
}
void serve_a(int i) {
   atomic_c += 1;
   atomic_c2 += atomic_c;
}
void serve_am(int i) {
   lock_guard l(m);
   serve_a(i);
}
```

kod	raw_c		raw_c2		atomic_c		atomic_c2	
	min	max	min	max	min	max	min	max
`thread th1( serve, 1 ); thread th2( serve, 2 ); th1.join(); th2.join();`								
`thread th1( serve_m, 1 ); thread th2( serve_m, 2 ); th1.join(); th2.join();`								
`thread th1( serve_a, 1 ); thread th2( serve_a, 2 ); th1.join(); th2.join();`								
`thread th1( serve_am, 1 ); thread th2( serve_am, 2 ); th1.join(); th2.join();`								

Zadanie 2 (1pkt)

Podaj wydruk. Liczba liter Twojego nazwiska (stała NAME_SIZE jest użyta w zadaniu):

`const int NAME_SIZE =` [ ];

```
using PF = std::function<int (int)>;
vector<PF> vpf;
vpf.push_back([](int x){ return x; });
vpf.push_back([](int x){ return 2*x; });
vpf.push_back([](int x){ return 3*x; });
vpf.push_back([](int x){ return 4*x; });

int count = 0;
for_each(vpf.begin(), vpf.end(),[&](PF f){ count += f(NAME_SIZE);});
cout << count << endl;
```

Zadanie 3 (1pkt)
Podaj wydruk.

```
using Graph = boost::adjacency_list<boost::vecS, boost::vecS, boost::bidirectionalS>;
using Edge = pair<int, int>;
   enum { A, B, C, D, E, F, NUM_VERTICES };
const vector<Edge> EDGES =
   { Edge(A,B), Edge(A,C), Edge(A,D), Edge(A,E), Edge(E,F), Edge(F,E) };
Graph g(EDGES.begin(), EDGES.end(), NUM_VERTICES);
for_each( vertices(g).first, vertices(g).second,
        [&](Graph::vertex_descriptor v){
           for_each(out_edges(v, g).first, out_edges(v, g).second,
                  [&](Graph::edge_descriptor e){ cout << target(e, g) << '␣';});
        });
        cout << endl;
```

Pytanie 1 (1pkt)
Jakie treści proponujesz dodać do ZPR w przyszłych edycjach?

Jakie treści proponujesz usunąć z ZPR w przyszłych edycjach?

Pytanie 2 (1pkt)

Ile godzin poświęciłeś na wykonanie projektu z ZPR?

Ile godzin poświęciłeś na pozostałe elementy przedmiotu (wykłady, kolokwia, zadanie dodatkowe, itd.)?

Uwagi do prowadzącego (R.Nowak):

Imię i nazwisko: | | | | | | | | | | | | | | | | | | | | | | | | | | **23Z-2** 60 min

Zadanie 4 (2pkt)

Dane są klasy `Point`, `Rectangle`, `Triangle` oraz funkcja `distance`. Napisz funkcję `maxPerimeter`, która zwraca figurę o największym obwodzie. Sygnatura tej funkcji jest podana w rozwiązaniu. Szablon `std::span` reprezentuje sekwencję obiektów, ma m.in. metody `begin`, `end`.

```
struct Point {
   int x;
   int y;
};
struct Rectangle {
   Point topLeft;
   Point bottomRight;
};
struct Triangle {
   Point vertices[3];
};

using Shape = std::variant<Rectangle, Triangle>;

std::floating_point auto distance(Point from, Point to) {
   const auto dist_x = to.x - from.x;
   const auto dist_y = to.y - from.y;
   return std::sqrt(dist_x*dist_x + dist_y*dist_y);
}
```

```
auto maxPerimeter(std::span<Shape> shapes)
```

Zadanie 5 (2pkt)

Korzystając z algorytmów biblioteki standardowej napisz funkcji `how_many_odd`, która przyjmuje zestaw słów (jako parę iteratorów) i zwraca liczbę słów, których każdy znak jest liczbą parzystą (tzn. kod znaku jest liczbą parzystą).

```
std::integral auto how_many_odd(auto from, auto to);
```

Uwagi do prowadzącego (W. Wysota):