Imię i nazwisko:..... **A2** czas: 60 min

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

Zadanie 1 (6pkt)

Obiekt typu `Queue` przechowuje dane, które wstawia tam jeden wątek (kodu nie zamieszczono). Ilość danych w kolejce rośnie. Aby temu zapobiec zwiększono ilość wątków przetwarzających dane, które wykonują funkcję `calc_thread`. Niestety nie obserwuje się poprawy, a niektóre rdzenie procesora nie są obciążone. Popraw kod wątków przetwarzających dane.

```
class Data {}; //nie zamieszczono skladowych
using PData = std::shared_ptr<Data>;
class Queue {
public:
  bool empty() { return q_.empty(); }
  PData get() {
    PData out;
    if(!q_.empty()) {
       out = q_.front();
       q_.pop();
     }
    return out;
  }
  void put(PData data) { q_.push(data); }
  mutable boost::mutex m_;
private:
  std::queue<PData> q_;
public:
};
void calculate(const Data& d) {
  //długotrwałe obliczenia
}
void zad1() {
  Queue q;
  //pominięto kod wątków, które wstawiają
obiekty PData
  boost::thread_group thrds;
  for (int i=0; i < 10; ++i)
    thrds.create_thread(boost::bind(&calc_thrd, boost::ref(q)));
  thrds.join_all();
}
```

```
void calc_thrd(Queue& q) {
   while(!q.empty()) {
      boost::mutex::scoped_lock l(q.m_);
      if(!q.empty()) {
         calculate(*q.get());
      }
   }
}
```

Zadanie 2 (6pkt)

1) Zaimplementować szablon **`dot`**, który obliczy iloczyn kartezjański dwóch wektorów liczb, tzn. zwróci wartość $x_0y_0 + x_1y_1 + ... + x_{n-1}y_{n-1}$. Przyjąć, że wektory x i y są reprezentowane przez `std::array<T,N>`, gdzie T jest typem liczbowym, zaś N jest rozmiarem.

2) Postaraj się, aby szablon nie używał pętli do obliczeń. N będzie małe.

3) Zadbaj, aby można było wołać szablon tylko dla typów numerycznych.

`template <class T> struct is_arithmetic` bada, czy typ jest typem numerycznym, jeżeli tak, to `is_arithmetic<T>::value` jest typu `bool` i ma wartość `true`, w przeciwnym wypadku `is_arithmetic<T>::value` ma wartość `false`.

Notatki / uwagi do prowadzącego

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`.

```
template <unsigned n> int fun(int k) { return n + fun<n-1>(k+1); }
template <> int fun<0>(int k) { return k; }

void zad3() {
   cout << fun<1>(NAME.size()) << "␣" << fun<2>(NAME.size()) << "␣" << fun<5>(NAME.size()) << endl;
}
```

Zadanie 4 (4pkt)

Podaj napis, który zostanie wydrukowany przez funkcję `zad4`.

```
void zad4() {
   using Graph = boost::adjacency_list<boost::vecS, boost::vecS, boost::bidirectionalS>;
   using Edge = pair<int, int>;
   enum { A, B, C, D, E, F, G, NUM_VERTICES };
   const vector<Edge> EDGES =
      { Edge(A,B), Edge(A,C),
        Edge(A,D), Edge(A,E),
        Edge(E,F), Edge(F,G) };
   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;
}
```

Zadanie 5 (4pkt)

Dostarcz funkcję `max_letter`, która analizuje wektor napisów i zwraca ile razy występuje najczęstsza litera. Jeżeli kolekcja jest pusta zwracamy zero. Przykładowo `max_letter(vector<string>{"ALA","ELA","ALA"})` zwraca 5. Użyj algorytmów z biblioteki standardowej.

```
int max_letter(const std::vector<string>& words)
```

Pytanie 1 (1pkt)

Ile godzin w semestrze poświęciłeś na przedmiot ZPR (w sumie, tzn. obecność na wykładzie + nauka własna + projekt + kolokwia)

Pytanie 2 (1pkt)

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