Imię i nazwisko: | | | | | | | | | | | | | | | | | | | | | | | | | | **24Z** 60 min

Przyjąć, że udostępniona jest przestrzeń nazw std i są dodane niezbędne nagłówki z biblioteki standardowej

Zadanie 1 (2pkt)

Obiekt typu `Advert` przechowuje i zarządza życiem obiektów `Image, Audio, Text`. Zmodyfikuj klasę `Advert`, m.in. dostarcz poprawny konstruktor, destruktor, konstruktor przenoszący. Uwzględnij wyjątki, mogą być zgłaszane w konstruktorach. Kopiowanie obiektów `Advert` jest zabronione i tak ma zostać.

```
class Image { //obiekt jest 'duzy', przechowuje obrazek
public: Image(int id);
};
class Audio { //obiekt jest 'duzy', przechowuje nagranie
public: Audio(int id);
};
class Text { //obiekt jest 'duzy', przechowuje tekst
public: Text(const string& t);
};
```

```
class Advert {
public:
   Advert(const string& txt, int image_id, int audio_id)
      : text_(nullptr),
        image_(nullptr),
        audio_(nullptr)
   {
   }
   ~Advert()
   {
   }

   Advert(Advert&& a) : text_(a.text_), image_(a.image_), audio_(a.audio_)
   {
   }

   Advert(const Advert& ) = delete;
   Advert& operator=(const Advert& ) = delete;
   Advert& operator=(Advert&&) = delete;
private:
   Text* text_;
   Image* image_;
   Audio* audio_;
};
```

Zadanie 2 (2pkt)

Opisz fragment aplikacji tworzonej obiektowo, w którym warto stosować jednocześnie C++ i Python. Fragment powinien zawierać minimum 3 klasy. Opisz odpowiedzialności klas i uzasadnij dlaczego uważasz, że powinna być dotarczona w C++ albo w Python. Skrócony opis odpowiedzialności podsystemu: 1-2 zdania:

Nazwa klasy, odpowiedzialność, język programowania, uzasadnienie wyboru:

Zadanie 3 (2pkt)

Mapa drogowa jest reprezentowana przez graf pokazany niżej. Jeżeli trzeba, to popraw implementację, aby poprawnie zarządzać powoływaniem i zwalanianiem obiektów.

```
class City; using PCity = shared_ptr<City>; using WCity = weak_ptr<City>; using UCity = unique_ptr<City>;
class Road; using PRoad = shared_ptr<Road>; using WRoad = weak_ptr<Road>; using URoad = unique_ptr<Road>;

int main() {
   Map map;
   PCity a = map.addCity("a");
   PCity b = map.addCity("b");
   map.addTwoWayRoad(a,b);
   return 0;
}
```

```
class City {
public:
   City(const std::string&it n) : name_(n) {}
   void addRoad( PRoad road) { roadsTo_.push_back(road); }
private:
   std::string name_;
   vector<PRoad> roadsTo_;
};
class Road {
public:
   Road(PCity from, PCity to) : from_(from), to_(to) { }
private:
   PCity from_;
   PCity to_;
};

class Map {
public:
   Map() {}
   PCity addCity(const std::string& name) {
      PCity city = make_shared<City>(name);
      cities_.push_back(city);
      return city;
   }
   void addTwoWayRoad(PCity from, PCity to) {
      addOneWayRoad(from, to);
      addOneWayRoad(to, from);
   }
   void addOneWayRoad(PCity from, PCity to) {
      PRoad r = make_shared<Road>(from, to);
      from->addRoad(r);
   }
private:
   vector<PCity> cities_;
  };
```

Uwagi do prowadzącego (R.Nowak):

Imię i nazwisko: | | | | | | | | | | | | | | | | | | | | | | | | | | **24Z** 60 min

Zadanie 4 (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.

```
#include <algorithm>
#include <iostream>
#include <string>
#include <thread>
#include <unordered_map>
#include <vector>

typedef std::unordered_map<char, int> letter_counts;
typedef std::vector<std::string> strings;

letter_counts count_letters(const strings& texts) {
   letter_counts counts;
   for (const auto& text : texts) {
      for (const auto& letter : text) {
         ++counts[letter];
      }
   }
   return counts;
}

void uppercasify(strings& values) {
   for(auto& s : values){
      std::transform(s.begin(), s.end(), s.begin(),
         [](char c){ return std::toupper(c); });
   }
}

letter_counts uppercasify_and_count_letters(strings& values) {
   letter_counts l;

   std::thread t2([&l, &values]() { l = count_letters(values); });
   std::thread t1([&values]() { uppercasify(values); });
   t2.join();
   t1.join();

   return l;
}

int main() {
   strings values = {"Alice", "Crocodile", "Is␣water␣wet?"};
   auto mean_median = uppercasify_and_count_letters(values);
   for (const auto& letter : mean_median) {
      std::cout << letter.first << "␣" << letter.second << std::endl;
   }
   return 0;
}
```

Zadanie 5 (2pkt)

Czy przy zrównoleglaniu algorytmu typu ‘embarrassingly parallel‘ istnieje potrzeba wykorzystania jakiegoś mechanizmu synchronizacji? Podaj przykład takiego algorytmu i na jego podstawie uzasadnij swoją odpowiedź.

Pytanie (0 pkt)

Czy uważasz, że woda jest mokra? Odpowiedz Tak lub Nie. Pytanie nie posiada poprawnej/preferowanej odpowiedzi i istnieje w celu zebrania statystki.

Uwagi do prowadzącego (Ł. Neumanna):