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

Przyjąć, że udostępniona jest przestrzeń nazw std

Zadanie 1 (2pkt)

Obiekty typu `HRSystem` przechowuje i zarządza życiem obiektów typu `Staff`. Zmień kod, aby aby składowa `staff_` przechowywała sprytne wskaźniki typu `unique_ptr` (zmiana na: `vector<UStaff> staff_`) i całość działała poprawnie (np. obiekty `Manager`).

Opisz wybraną zaletę tej zmiany.

```
using Id = int;
using PStaff = shared_ptr<Staff>; using UStaff = unique_ptr<Staff>;
class HRSystem {
public:
  HRSystem() : staffId_(0) {}
  Id addEmployee() { //nowy pracownik etatowy, zwraca Id
    Id id = ++staffId_;
    staff_.push_back(make_shared<Employee>(*this, id) );
    return id;
  }
  Id addManager() { //nowy kierownik, zwraca Id
    Id id = ++staffId_;
    staff_.push_back(make_shared<Manager>(*this, id) );
    return id;
  }
private:
  int staffId_; //do generowania id, funkcja pomocnicza
  vector<PStaff> staff_;
};
class Staff {
public:
   Staff(HRSystem& owner, Id id) : owner_(owner), id_(id){}
   virtual ~Staff() {}
   virtual void addStaff(PStaff s) {} //niepusta tylko dla Managera
protected:
   HRSystem& owner_;
   Id id_;
};
class Employee : public Staff {
public:
   Employee(HRSystem& owner, Id id) : Staff(owner, id){}
};
class Manager : public Staff {
public:
   Manager(HRSystem& owner, Id id) : Staff(owner, id){}
   void addStaff(PStaff s) { staff_.push_back(s); }
private:
   vector<PStaff> staff_; //podwladni
};
```

Zadanie 3 (2pkt)

Zaimplementuj funkcję `median`, zwraca średnią wartość dla 3 liczb. Dla przykładów obok zawsze zwraca 2. Wykorzystaj funkcje standardowe `std::min(x,y)` oraz `std::max(x,y)` które zwracają odpowiednio mniejszą i większą wartość argumentu.

```
median(1,2,3);
median(1,3,2);
median(2,1,3);
median(2,3,1);
median(3,1,2);
median(3,2,1);
```

```
   int median(int a, int b, int c) {
```

Zadanie 2 (2pkt)

Pokazane 4 klasy, w hierarchii `Staff` mają powielony kod. Postaraj się to zminimalizować. Nie zmieniamy klasy `HRSystem`. Całość reprezentuje osoby zatrudnione w przedsiębiorstwie i rejestruje każde żądanie obliczenia premii. Premia dla pracowników etatowych (`Employee`) jest zależna od pensji i stażu, premia dla kierowników to dodatkowa pensja, nie przewiduje się premii dla stażystów.

```
using Money = int;
using Id = int;
using UStaff = unique_ptr<Staff>;
class HRSystem {
public:
  HRSystem() : staffId_(0) {}
  void noticeBonus(Id id) { bonuses_.push_back(id); }
  Id addEmployee(Money salary, int seniority) { //nowy pracownik etatowy
    Id id = ++staffId_;
    staff_.push_back(make_unique<Employee>(*this, id, salary, seniority) );
    return id;
  }
  Id addTrainee() { //nowy stazysta
    Id id = ++staffId_;
    staff_.push_back(make_unique<Trainee>(*this, id) );
    return id;
  }
  Id addManager(Money salary, int seniority) { //nowy kierownik
    Id id = ++staffId_;
    staff_.push_back(make_unique<Manager>(*this, id, salary, seniority) );
    return id;
  }
private:
  int staffId_; //do generowania identyfikatorow
  vector<UStaff> staff_;
  vector<Id> bonuses_;
};
```

```
class Staff {
public:
   Staff(HRSystem& owner, Id id) : owner_(owner), id_(id) {}
   virtual Money calcBonus() const = 0;
   virtual ~Staff() {}
protected:
   HRSystem& owner_;
   Id id_;
};
class Employee : public Staff {
public:
   Employee(HRSystem& own, Id id, Money sal, int sen) : Staff(own,id), salary_(sal), seniority_(sen){}
   virtual Money calcBonus() const {
      owner_.noticeBonus(id_);
      return salary_ * seniority_ / 100;
   }
private:
   Money salary_;
   int seniority_; //lata pracy
};
class Trainee : public Staff {
public:
   Trainee(HRSystem& own, Id id) : Staff(own,id){}
   virtual Money calcBonus() const {
      owner_.noticeBonus(id_);
      return 0; //no bonus for Trainee
   }
};
class Manager : public Staff {
public:
   Manager(HRSystem& own, Id id, Money sal, int sen) : Staff(own,id), salary_(sal), seniority_(sen){}
   virtual Money calcBonus() const {
      owner_.noticeBonus(id_);
      return salary_;
   }
private:
   Money salary_;
   int seniority_; //lata pracy
};
```

Uwagi do prowadzącego (R.Nowak):

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

Zadanie 4 (2pkt)

Czy każdy rodzaj wyścigów może być wyeliminowany na poziomie kompilacji kodu? Odpowiedź uzasadnij.

Zadanie 5 (2pkt)

Czy automatyczne (implicit) konwersje między typami numerycznymi podczas operacji arytmetycznych w C++ mogą powodować problemy? Odpowiedź uzasadnij.

Uwagi do prowadzącego (Ł. Neumanna):