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

Przyjąć, że udostępniona jest przestrzeń nazw std

Zadanie 1 (5pkt)

W edytorze reprezentujemy Elementy jako obiekty klasy `Paragraph`, `List` lub `Name`. Zaimplementuj funkcję `read_statistics`. Przykład użycia (`zad1`) drukuje `2,1,2`.

```
using cstr = const string&;
class Element {
public:
   virtual void accept(Visitor& v) const = 0;
};
class Visitor {
public:
   virtual void visit(const Paragraph& p) = 0;
   virtual void visit(const List& l) = 0;
   virtual void visit (const Name& t) = 0;
};
void zad1() {
   Paragraph p1("XXX"), p2("YYYY");
   List l1;
   Name n1("Adam", "Abacki"), n2("Jing", "Jang");
   vector<Element*> v = { &p1, &l1, &p2, &n1, &n2 };
   Statistics s = read_statistics(v);
   cout << s.p_ << "," << s.l_ << "," << s.n_ << endl;
}
```

```
class Paragraph : public Element {
public:
   Paragraph(cstr s) : str_(s) {}
   virtual void accept(Visitor& v) const { v.visit(*this); }
   cstr get() const { return str_; }
private:
   string str_;
};
class List : public Element {
public:
   List() : Element(l) {}
   virtual void accept(Visitor& v) const { v.visit(*this); }
};
class Name : public Element {
public:
   Name(cstr f, cstr fam ) : first_(f), family_(fam) {}
   virtual void accept(Visitor& v) const { v.visit(*this); }
   cstr getFirst() const { return first_; }
   cstr getFamily() const { return family_; }
private:
   string first_;
   string family_;
};
```

```
struct Statistics {
   Statistics() : p_(0), l_(0), n_(0) {}
   int p_, l_, n_;
};

Statistics read_statistics(const vector<Element*>& v)
```

Zadanie 2 (7pkt)

Różne obiekty typu `Element` (patrz Zadanie 1) mogą reprezentować napisy w różnych językach. Dostarczamy hierarchię `Language`, zakładamy dla uproszczenia, że do reprezentacji znaku wystarczy `char`). Zaproponuj modyfikacje, aby to umożliwić. Przykładowe klasy konkretne `Language` pokazano na sąsiedniej stronie (ich implementacja nie jest tutaj istotna). Jakiego wzorca projektowego użyjesz?

```
   class Element
```

```
class Language {
public:
   virtual bool compare(char c1, char c2) = 0; //porzadek znakow
   virtual char digitSep() const = 0; //separator dziesietny
   virtual void printName(ostream& os, cstr& first, cstr& family) = 0; //pisze imie i nazwisko
};
using PLang = shared_ptr<Language>;
```

```
class Polish : public Language {
public:
   virtual bool compare(char c1, char c2); //implementacja nieistotna
   virtual char digitSep() const { return ',' ; }
   virtual void printName(ostream& os, cstr& first, cstr& family) { os << first << '␣' << family; }
};
class Chinese : public Language {
public:
   virtual bool compare(char c1, char c2) //implementacja nieistotna
   virtual char digitSep() const { return '.'; }
   virtual void printName(ostream& os, cstr& first, cstr& family) {
     string str(family); boost::to_upper(str); os << str << '␣' << first;
     //boost::to_upper zmienia kazda litere na wielka
   }
};
```

Zadanie 3 (3pkt)

Podaj napis generowany przez `zad3` [ ], NAME to stała typu `string` zawierająca Twoje nazwisko zapisane wielkimi literami ASCII (zamiast 'Ą' jest 'A'), np. `WROZKA` dla Wróżka.

```
class E
  : public std::exception
{};
class B {
public:
  B(int i) : i_(i) {
    cout << i_;
  }
  virtual ~B() {
    cout << i_;
  }
  int get() const {
    return i_;
  }
private:
  int i_;
};
```

```
class D1 : virtual public B {
public:
   D1(int i) : B(i+1) {}
};
class D2 : virtual public B {
public:
   D2(int i) : B(i+2) {}
};
class M : public D1, public D2 {
public:
   M(int i) : B(i),D1(i),D2(i) {}
   int get() const {
     return D1::get() + D2::get();
   }
};
```

```
void f(int i, const M& m) {
  if(m.get() < 0) throw E();
  g(i+3);
}
void g(int i) {
  const int N = NAME.size() % 3;
  M m(N-i);
  f(i, m);
}
void zad3() {
  try {
    g(1);
  } catch(E&){
  }
  cout << endl;
}
```

Zadanie 4 (5pkt)

Zmień implementację klas reprezentujących węzły, aby poprawnie przechowywać i zwalniać obiekty. Przykład użycia to funkcja `zad4`.

```
struct Node;
using PNode = shared_ptr<Node>;
using QNode = weak_ptr<Node>;
struct Node {
   Node() { }
   ~Node() {}
   PNode up_;
   PNode down_;
   PNode left_;
   PNode right_;
};

struct Grid {
   Grid() : topLeft_(new Node) { }
   ~Grid() { }
   PNode topLeft_;
};
```

```
void connectDown(PNode up, PNode down) {
   up->down_ = down;
   down->up_ = up;
}
void connectLeft(PNode right, PNode left) {
   right->left_ = left;
   left->right_ = right;

}

void zad4() {
   Grid g;

   connectLeft(g.topLeft_, PNode(new Node));
   connectDown(g.topLeft_, PNode(new Node));
   connectLeft(g.topLeft_->down_, PNode(new Node));
   connectDown(g.topLeft_->left_, g.topLeft_->down_->left_ );
};
```

Zadanie 5 (3pkt)

Popraw deklarację metody fabrycznej `load_widget`.

```
widget* load_widget( widget::id desired );
```

Pytanie 1 (2pkt)

Wymień 2 zalety dostarczania użytkownikowi kodu binarnego.

Wymień 2 wady takiego podejścia.

Uwagi do prowadzącego: