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

Przyjąć, że udostępniona jest przestrzeń nazw std

Zadanie 1 (5pkt)

Zmień implementację drzewa binarnego, aby poprawnie zwalniać obiekty. Przykład użycia to funkcja `zad1`.

```
class Tree;
using PTree = shared_ptr<Tree>;
using PWTree = weak_ptr<Tree>;
class Tree {
public:
   ~Tree() { }
   void addLeft(PTree left) {
      left_ = left;
      left_->up_ = me_.lock();
   }
   void addRight(PTree right) {
      right_ = right;
      right_->up_ = me_.lock();
   }
   static PTree create(const string& name); //funkcja fabryczna
private:
   Tree(const string& name) : name_(name) { }
   string name_;
   PWTree me_;
   PTree up_;
   PTree left_;
   PTree right_;
};
```

```
PTree Tree::create(const string& name) {
   PTree node(new Tree(name) );
   node->me_ = node;
   return node;
}

void zad1() {
   PTree root = Tree::create("root");
   root->addLeft( Tree::create("left") );
}
```

Zadanie 2 (3pkt)

Podaj napis generowany przez `zad2` [ ], NAME to stała typu `std::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 doGet();
  }
private:
  virtual int doGet() const {
    return i_;
  }
  int i_;
};
```

```
class D1 : virtual public B {
public:
  D1(int i) : B(i) {}
private:
  virtual int doGet() const {
    return 0;
  }
};
class D2 : virtual public B {
public:
  D2(int i) : B(i) {}
};
class M : public D1, public D2 {
public:
  M(int i) : B(i),D1(i),D2(i) {}
};
```

```
void f(int i, const M& m) {
  if(i <= 0) throw E();
  g(m.get());
}
void g(int i) {
  const int N = NAME.size() % 4 + 4;
  M m(N);
  f(i, m);
}
void zad2() {
  try {
    g(1);
  } catch(E&){
  }
  cout << endl;
}
```

Zadanie 3 (3pkt)

Popraw kod, metoda `isBest` działa poprawnie, ale nie została napisana optymalnie.

```
class Foo {
public:
   explicit Foo(int val = 0) : val_(val) {}
   bool isBest() const {
      if(!(val_ >= 0) || !(val_ <= 0))
         return true;
      else if ( (val_ < -1) && (val_ > 1) )
         return true;
      else if ( val_ == 1 )
         return true;
      return false;
   }
private:
   int val_;
};
```

Pytanie 1 (1pkt)

Dlaczego używa się różnych języków programowania do tworzenia tej samej aplikacji ?

Zadanie 4 (6pkt)

Napisz wizytator, który obliczy wartość wyrażenia reprezentowaną przez kolekcję obiektów `Element`. Przykład użycia to funkcja `zad4`.

```
class Element {
public:
  virtual void accept(Visitor& v) = 0;
};
class Visitor {
public:
  virtual ~Visitor() {}
  virtual void visit(Number&) = 0;
  virtual void visit(OperatorAdd&) = 0;
};
class Number : public Element {
public:
   Number(int val) : val_(val) {}
   int getVal(){ return val_; }
   virtual void accept(Visitor& v) { v.visit(*this); }
private:
   int val_;
};
class OperatorAdd : public Element {
public:
   OperatorAdd(){}
   virtual void accept(Visitor& v) { v.visit(*this); }
};
```

```
class Calculator : public Visitor {
```

```
void zad4() {
   using PElement = shared_ptr<Element>;
   vector<PElement> v;
   v.push_back(PElement(new Number(1))); v.push_back(PElement(new Number(2))); v.push_back(PElement(new OperatorAdd()));
   v.push_back(PElement(new Number(3))); v.push_back(PElement(new OperatorAdd()));

   Calculator c;
   for(PElement e : v) { e->accept(c); }
   cout << c.value() << endl;
}
```

Zadanie 5 (7pkt)

(1) Zaproponuj klasy do reprezentacji żołnierza (`Soldier`) i dowódcy (`Leader`), aby można było tworzyć drzewiaste struktury dowodzenia, dowódca może dowodzić żołnierzami lub innymi dowódcami. (2) Dostarcz metodę do tworzenia głębokiej kopii drzewiastej struktury obiektów.

Uwagi do prowadzącego: