Imię i nazwisko: ..... | T | czas: 60 min

Przyjąć, że udostępniona jest przestrzeń nazw std

Zadanie 1 (5pkt)

Elementy listy (typu `Node`) nie są zwalniane w momencie niszczenia obiektu `List`. Popraw implementację.

```
class List {
   struct Node;
   typedef std::shared_ptr<Node> PNode;
   typedef std::weak_ptr<Node> PWNode;
public:
   List() {}
   ~List() {}
   void push_front(int val);
private:
   PNode head_;
   PNode head2_;
};
```

```
struct List::Node {
  Node(int v = 0) : val_(v) {}
  ~Node() { }
  int val_;
  PNode next_, next2_, prev_;
};
void List::push_front(int val) {
   PNode n(new Node(val));
   n->next_ = head_;
   n->next2_ = head2_;
   if(head_)
      head_->prev_ = n;
   head2_ = head_;
   head_ = n;
}
```

Zadanie 2 (4pkt)

Liczba liter Twojego nazwiska `LICZBA_LITER=` [ ]. Podaj napis generowany przez `zad2` [ ]

```
class E : public std::exception {
public:
  E(int i) : i_(i) {}
  int i_;
};
struct F {
  F(int i) : i_(i) {cout << i_ << ',';}
  virtual ~F() {cout << i_ << ',';}
  int i_;
};
```

```
typedef auto_ptr<F> PF;

void f(PF pf) {
   if(pf->i_ > 2) {
      f(PF(new F(pf->i_ / 2)));
   }
   else {
      throw E(pf->i_);
   }
}
```

```
void zad2() {
  try {
    PF pb(new F(LICZBA_LITER));
    f(pb);
  } catch(E& e){
    cout << e.i_;
  }
  cout << endl;
}
```

Zadanie 3 (3pkt)

Podaj napis generowany przez `zad3` [ ]

```
class B {
  int i_;
public:
  B(int i) : i_(i) {}
  virtual void inc() {++i_;}
  int get() {return i_;}
};
class D1 : virtual public B {
public:
  D1(int i) : B(i-1) {}
  virtual void inc() {
    B::inc(); B::inc();
  }
};
```

```
class D2 : virtual public B {
public:
  D2(int i) : B(i-2) {}
  virtual void inc(){
    B::inc();
  }
};
class M : public D1, public D2 {
public:
  M(int i) : B(i), D1(i), D2(i) {}
  virtual void inc() {
    D1::inc(); D2::inc();
  }
};
```

```
void zad3() {
   M m(3);
   cout << m.get();
   m.inc();
   cout << m.get();
   static_cast<D1&>(m).inc();
   cout << m.get();
   cout << endl;
}
```

Pytanie 1 (1pkt)

Dlaczego kod źródłowy powinien być czytelny ?

Pytanie 2 (1pkt)

Jakie informacje umieszcza się w komentarzach ?

Uwagi do prowadzącego:

Zadanie 4 (6pkt)

Uzupełnij implementację klasy `Button`, która reprezentuje przycisk, metoda `click` jest wołana w momencie wciśnięcia. Przycisk powinien, w zależności od argumentu użytego podczas konstrukcji, włączać bądź wyłączać światło (obiekt typu `LightSwitch`) lub wentylator (obiekt typu `FanSwitch`). Proszę opisać w jaki sposób, w zaproponowanym rozwiązaniu można przechowywać historię wykonywanych akcji.

```
class LightSwitch {
public:
   void lightOn();
   void lightOff();
};
class FanSwitch {
public:
   void turnUp();
   void turnDown();
};
```

```
class Button {
public:
   void click() {
```

Zadanie 5 (5pkt)

Pracownicy etatowi (`MonthlyEmployee`) pracują 160 godzin miesięcznie, zaś inni (`HourlyEmployee`) przechowują liczbę przepracowanych godzin. Uzupełnij implementację funkcji `countHours`, aby zliczać liczbę przepracowanych godzin dla kolekcji obiektów `Employee`. Przykład użycia to funkcja `zad5`.

```
class Employee {
public:
   virtual void accept(Visitor& v)=0;
};
class Visitor {
public:
   virtual ~Visitor() {}
   virtual void visit(HourlyEmployee&)=0;
   virtual void visit(MonthlyEmployee&)=0;
};
```

```
class HourlyEmployee : public Employee {
public:
   HourlyEmployee(int hours) : hours_(hours) {}
   int getHours() const { return hours_; }
   virtual void accept(Visitor& v){v.visit(*this);}
private:
   int hours_;
};
class MonthlyEmployee : public Employee {
 public:
   static const int HOURS_IN_MONTH = 160;
   virtual void accept(Visitor& v){v.visit(*this);}
};
void zad5() {
   HourlyEmployee he1(12), he2(8);
   MonthlyEmployee me1;
   vector<Employee*> v;
   v.push_back(&he1);
   v.push_back(&he2);
   v.push_back(&me1);
   cout << countHours(v) << endl;
}
```

```
   int countHours(vector<Employee*> employee)
```