Imię i nazwisko: | | | | | | | | | | | | | | | | | | | | | | | | | | **25Z-1** 60 min

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

Zadanie 1 (2pkt)

`Node` reprezentuje węzeł. Węzły w liście jednokierunkowej są powiązane sprytnymi wskaźnikami, aby ułatwić zwalnianie zasobów (składowa `next_`). Dodatkowo tworzone są listy jednokierunkowe dla elementów o tej samej wartości (składowa `next_eq_`). Jeżeli trzeba, to popraw kod, aby zwalnianie elementów było poprawne.

```
class List {
   using PNode = std::shared_ptr<Node>;
   using PWNode = std::weak_ptr<Node>;
   struct Node : public std::enable_shared_from_this<Node> {
      Node(int v) : value_(v) {}
      PNode getptr() {
         return shared_from_this();
      }
      ~Node() {}
      int value_;

      PNode next_; //element nastepny
      PNode next_eq_; //element o tej samej wartosci
   };
public:
   List() {}
   ~List() {}
   void push_front(int value) {
      PNode node = make_shared<Node>(value);
      node->next_ = head_;
      head_ = node;
      for(PNode n = head_->next_; n != nullptr; n = n->next_) {
         if(n->value_ == value) {
            node->next_eq_ = n;
            break;
         }
      }
   }
private:
   PNode head_;
};
```

Zadanie 2 (2pkt)

Obiekty typu `Point` reprezentują punkty wielowymiarowe, dlatego składową jest wektor liczb. Nie można zmieniać tej składowej. Popraw kod, aby uniknąć niepotrzebnego kopiowania wektorów, np. w ostatniej linii w przykładzie obok. Uwaga, proszę pamiętać o funkcji `std::move`, poniżej dokumentacja

`A = std::move(B);` Now A contains the elements that were previously held by B, and B is now empty. This avoids copying: the internal representation is simply moved from B to A, so this is an O(1) solution.

```
Point h(Point p) {
  p.inc_x();
  //dalszy kod nieistotny dla zadania
  return p;
}
 Point p1(1.0, 1.0);
 h(p1);

 h(h(h(Point(2.0, 2.0))));
```

```
class Point {
   vector<double> vec_;
public:
   Point(double x, double y) {
      vec_.push_back(x);
      vec_.push_back(y);
   }
   Point(const Point& p) {
   }
   Point(Point&& p) {
   }
   void inc_x() {
      vec_[0] += 1.0;
   }
   ~Point() { }
};
```

Zadanie 3 (2pkt)

Obiekty typu `Record` można zapisać (w pliku na dysku, w strumieniu danych lub w innym medium - w zależności od konkretnej klasy. Zapis ma sens gdy rekord nie jest pusty oraz gdy został zmieniony. Zmień strukturę, aby unikać powielania kodu.

```
//klasa bazowa
class Record {
public:
   Record() {}
   virtual ~Record() = default;
   virtual void save() = 0;
   bool isEmpty() const { return false; } //details not required in this task!
   bool isModified() const { return true; } //details not required in this task!
};
class FileRecord : public Record {
public:
  FileRecord(const std::string& filename) {
    //media preparation, details not relevant to this task
  }
  void save() override {
    if( ! isEmpty() && isModified() ) {
      //saving implementation, details not relevant to this task
    }
  }
};
class StreamRecord : public Record {
public:
   StreamRecord(const std::string& stream_id) {
    //media preparation, details not relevant to this task
   }
   void save() override {
     if( ! isEmpty() && isModified() ) {
       //saving implementation, details not relevant to this task
     }
   }
};
```

Zadanie 4 (1pkt)

Daj przykład aplikacji/systemu, w którym warto stosować jednocześnie C++ i Python.
Skrócony opis systemu: (kilka słów)

Jakie są zadania dla modułów w C++:

Jakie są zadania dla modułów w Python:

Uwagi do prowadzącego (R.Nowak):

Zadanie 5 (2pkt)

Następujący kod skutkuje pętlą nieskończoną. Wytłumacz dlaczego tak się dzieje. Napraw kod tak, aby pętla wykonała się 100 razy.

```
#include <cstdint>
#include <print>

int main() {
   int16_t i {};
   unsigned int n = 99;
   for (i = 0; n - i >= 0; ++i) {
      std::println("Hi␣{}!", i);
   }
   return 0;
}
```

Zadanie 6 (2pkt)

Czym jest monomorfizacja w kontekście programowania generycznego/szablonów?

Uwagi do prowadzącego (Ł. Neumanna):

Notatki