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

Przyjąć, że udostępniona jest przestrzeń nazw std

Zadanie 1 (2pkt)

Graf jest reprezentowany przez 2 kolekcje: kolekcję wierzchołków i kolekcę krawędzi. Metoda 'delVertex' nie zawsze poprawnie usuwa wierzchołek, który jest argumentem Przykład błędnego działania (spodziewamy się usunięcia wierzchołka) jest pokazany poniżej. Popraw kod.

```
int main() {
  Graph g;
  {
    PVertex v1 = g.newVertex("A");
    PVertex v2 = g.newVertex("B");
    PEdge e = g.newEdge(v1, v2, 10);
    g.delVertex(v1);
  } //nie usuwa wierzcholka 'A', przy wyjsciu z bloku
  return 0;
}
```

```
struct Edge;
using PEdge = shared_ptr<Edge>;
using Edges = vector<PEdge>;
struct Vertex;
using PVertex = shared_ptr<Vertex>;
using Vertices = vector<PVertex>;

struct Edge {
   Edge(PVertex in, PVertex out, int len) : in_(in), out_(out), len_(len) { }
   ~Edge() { }
   PVertex in_;
   PVertex out_;
   int len_;
};
struct Vertex {
   Vertex(const string& name) : name_(name) { }
   ~Vertex(){ }
   string name_;
};
struct Graph {
   Edges edges;
   Vertices vertices;
   PVertex newVertex(const string& name) {
      PVertex v(new Vertex(name));
      vertices.push_back(v);
      return v;
   }
   PEdge newEdge(PVertex in, PVertex out, int len) {
      PEdge e(new Edge(in,out,len) );
      edges.push_back(e);
      return e;
   }
   void delVertex(PVertex v) {
      for(Vertices::iterator i = vertices.begin(); i != vertices.end(); ) {
         if(*i == v) {
            i = vertices.erase(i);
         }
         else {
            ++i;
         }
      }
   }
};
```

Zadanie 2 (2pkt)

Popraw funkcję calc, aby działała poprawnie uwzględniając możliwość zgłaszania wyjątków.

```
class Foo { //  klasa przykładowa, proszę jej nie modyfikować
public:
   Foo(const string& s) : s_(s) {}
   int val() const { return s_.size(); }
private:
   string s_;
};
class Goo { //  klasa przykładowa, proszę jej nie modyfikować
public:
   Goo(int v) : v_(v) {}
   int val() const { return v_; }
private:
   int v_;
};
int add(const Foo& f, const Goo& g) { // funkcja przykładowa, proszę jej nie modyfikować
   return f.val() + g.val();
}
```

```
Foo* createFoo(const string& s) {
   return new Foo(s);
}

Goo* createGoo(int i) {
   return new Goo(i);
}


int calc() {
   Foo* f = createFoo("x");
   Goo* g = createGoo(1);

   int out = add(*f, *g);

   delete f;
   delete g;

   return out;
}
```

Zadanie 3 (2pkt)

Klasy w hierarchii 'TransformSamples' przetwarzają kolekcję wartości (próbek, wektorów liczb rzeczywistych). Popraw implementację aby wyeliminować powtórzenia. Pamiętaj o wzorcu NVI.

```
using Samples = vector<double>;
//  Nie zmienia kolekcji.
class TransformSamples {
public:
   TransformSamples() { }
   virtual Samples transform(const Samples& in) {
      return Samples(in);
   }
};









//  Usuwa wartości ekstremalne: większe niż MAX_FACTOR * średnia, oraz mniejsze niż MIN_FACTOR * średnia.
class CutExtrema : public TransformSamples {
public:
   CutExtrema(double min_factor, double max_factor) : MIN_FACTOR(min_factor), MAX_FACTOR(max_factor) {}
   virtual Samples transform(const Samples& in) {
      if(in.empty() ) return Samples();
      Samples out;
      out.reserve(in.size());
      double avg = accumulate(in.begin(), in.end(), 0.0) / static_cast<double>(in.size());
      for(double d : in ) {
         if(d > MAX_FACTOR * avg ) out.push_back(MAX_FACTOR * avg);
         else if (d < MIN_FACTOR * avg) out.push_back(MIN_FACTOR * avg);
         else out.push_back(d);
      }
      return out;
   }
private:
   const double MIN_FACTOR;
   const double MAX_FACTOR;
};
// Odejmuje wartość średnią od każdej wartości.
class DelAvg : public TransformSamples {
public:
   DelAvg() {}

   virtual Samples transform(const Samples& in) {
      if(in.empty() ) return Samples();
      Samples out;
      out.reserve(in.size());
      double avg = accumulate(in.begin(), in.end(), 0.0) / static_cast<double>(in.size());
      for(double d : in ) {
         out.push_back(d - avg);
      }
      return out;
   }
};
```

Uwagi do prowadzącego (R.Nowak):

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

Zadanie 4 (2pkt)

Napisz klasę, której obiekty przechowują kolekcję liczb całkowitych. Klasa pozwala dodawać liczbę do kolekcji, dostarcza metodę 'add'. Klasa ma metodę, która zwraca medianę liczb przechowywanych w klasie. Liczenie mediany uznajemy za czasochłonne, należy zadbać o odpowiedni moment wykonywania tej operacji. W szczególności chcemy uniknąć przeliczania jej na nowo za każdym razem, kiedy ktoś poprosi o zwrócenie mediany lub gdy zostanie dodana nowa liczba do obiektu.

Zwróć uwagę na stałość (const) oraz sposób przekazywania danych i wyników. Podpowiedź: jeżeli chodzi o stan wewnętrzny klasy - mediana może być policzona lub nie (optional). Uwaga: Sytuacją szczególną jest to, gdy w kolekcji nie ma żadnych wartości.

Zadanie 5 (2pkt)

Uprość poniższy kod, korzystając z algorytmów biblioteki standardowej.

```
std::optional<std::string> bar(const std::string& input) {
  bool all_lower = true;
  for(int i = 0; i < input.size(); ++i) {
    if (isupper(input[i])) {
      all_lower = false;
    }
  }
  if (!all_lower) {
    return std::nullopt;
  }
  const std::vector<char> vowels = {'a', 'e', 'i', 'o', 'u', 'y'};
  int num_of_vowels = 0;
  for (int i = 0; i < input.size(); ++i) {
    for(auto v: vowels) {
      if (input[i] == v) {
        num_of_vowels++;
        break;
      }
    }
  }
  std::string output;
  const auto size = input.size();
  for (int i = 1; i <= num_of_vowels; ++i) {
     output.push_back(input[size-i]);
  }
  return output;
}
```

Uwagi do prowadzącego (W. Wysota):