Egzamin z Programowania Obiektowego.
Informatyka. II rok. 25-01-00.

Zad. 1 (25 p.)

W skład sieci telefonicznej wchodzą centrale i stacje abonenckie (w dalszej części opisu utożsamiane z abonentami). Centrale dzielą się na lokalne, miejskie i krajowe. Każdy abonent ma unikatowy numer telefonu i jest podłączony do jednej centrali lokalnej. Numer telefonu składa się z 4 części:

- numeru abonenta w centrali lokalnej,
- numeru centrali lokalnej,
- numeru centrali miejskiej,
- numeru centrali krajowej.

Każda centrala lokalna jest podłączona do jakiejś centrali miejskiej i przechowuje informacje o wszystkich rozmowach wychodzących od podłączonych do niej abonentów (kto do kogo dzwonił i ile czasu trwało połączenie). Każda centrala miejska jest podłączona do jakiejś centrali krajowej. Każda centrala miejska wie w jakim jest mieście. Każda centrala krajowa zna swoje sąsiednie centrale krajowe (p. też opis realizacji żądania połączenia). Każda centrala zna swój numer.

Realizacja żądania połączenia: dzwoniący przekazuje do swojej centrali lokalnej numer docelowy, z którym się chce połączyć (może pominąć numer centrali miejskiej i krajowej lub sam numer centrali krajowej, o ile są takie same jak u niego). Jeśli numer docelowy jest w tej samej centrali lokalnej, to ta realizuje połączenie, wpp. centrala lokalna przekazuje żądanie połączenia do centrali miejskiej, ta postępuje analogicznie ewentualnie przekazując żądanie do centrali krajowej. Centrala krajowa pamięta dla każdej docelowej centrali krajowej, do której z sąsiednich central krajowych należy przekazać żądanie połączenia, by dotarło ono do celu. Docelowa centrala krajowa przekazuje żądanie do właściwej centrali miejskiej, ta do lokalnej, a ta z kolei do abonenta. Żądanie połączenia nie jest realizowane, jeśli podano zły numer (z nieistniejącym numerem centrali lub abonenta) lub jeśli abonent właśnie rozmawia.

Należy:

- Zaprojektować hierarchię klas opisującą powyżej zdefiniowaną sieć telefoniczną. (Opis każdej klasy musi zawierać jej zmienne/metody klasowe/instancyjne wraz z ich opisami.)
- Napisać metodę *zadzwonIRozmawiaj:* realizującą jedno połączenie telefoniczne abonenta. Treść tej metody powinna mieć następującą postać:
  1. żądanie połączenia - metoda *połącz: nrDocelowy*, dająca w wyniku *nil* (niezrealizowane żądanie) lub docelowego abonenta,
  2. przeprowadzenie rozmowy – metoda *rozmowa: abonent*, dająca w wyniku czas trwania rozmowy (treść tej metody w tym zadaniu nas nie interesuje),
  3. zakończenie rozmowy polegające na zapamiętaniu informacji statystycznych o rozmowie (kto do kogo i jak długo dzwonił) i zawiadomieniu abonenta docelowego, że rozmowa została zakończona.

  (Oczywiście punkty 2 i 3 dotyczą tylko sytuacji, gdy w punkcie 1 nawiązano połączenie.)
- Napisać metodę centrali lokalnej wypisującą na zadany strumień zebrane informacje o wszystkich zrealizowanych połączeniach wychodzących od abonentów podłączonych do tej centrali.
- Zaimplementować **wszystkie** niestandardowe metody (oprócz metody *rozmowa:*) użyte w rozwiązaniu zadania.

Zad. 2 (15 p.)

Dany jest wzorzec klasy BST (implementującej drzewo binarnych wyszukiwań):

```
template <class T>
class BST{
  // ...
 public:
  BST();
  BST(const BST&);
  const BST& operator=(const BST&);
  ~BST();
  BST& wstaw(const T&);          // wstawia do BST (także duplikaty)
  BST& usun(const T&);           // usuwa z BST (tylko jeden egzemplarz, o ile istnieje, wpp. błąd)
  BSTIter<T>* dajIterator();     // daje iterator po BST, ustawiony na minimalnym elemencie
};
```

i wzorzec klasy BSTIter (implementujący przechodzenie po elementach BST wg rosnących wartości):

```
template <class T>
class BSTIter{
  // ...
 public:
  // ...
  int jestElement();          // informuje czy jest jeszcze bieżący element
  T dajElement();             // daje bieżący element (o ile jestElement(), wpp. błąd)
  void nastepny();            // przechodzi do następnego (wg wielkości) elementu
};
```

Napisz wzorzec abstrakcyjnej klasy SortColl (opisującej kolekcję elementów jednego typu posortowaną wg relacji <=) oraz wzorzec klasy BSTSortColl implementującej posortowaną kolekcję jako drzewo binarnych wyszukiwań. Posortowane kolekcje powinny dostarczać operacje *wstaw, usun* (odpowiednio wstawiającą i usuwającą wskazany element, oczywiście w kolekcji posortowanej elementy mogą się powtarzać i nie można usuwać nieistniejących obiektów), *rozmiar* (dającą liczbę elementów w kolekcji) oraz *wypisz* (wypisującą wszystkie elementy kolekcji od najmniejszego do największego). Obiekty klasy BSTSortColl muszą się poprawnie kopiować, przypisywać, tworzyć (konstr. bezargumentowy) i niszczyć. Zaimplementuj odpowiednie metody/konstruktory/destruktory lub uzasadnij ich brak.

Powodzenia!