Zadatak:

Napišite sučelje i implementaciju za strukturu **`vlakovi`**. Struktura **`vlakovi`** sadrži mapu koja postojećim željezničkim direktnim linijama pridružuje pozitivnu cjelobrojnu cijenu. Pri tome je željeznička linija opisana parom stringova, različitim imenima krajnjih stanica. Cjenik je simetričan, tj. ako postoji linija od mjesta A do mjesta B sa cijenom c, onda postoji i linija od mjesta B do mjesta A sa istom cijenom. Sučelje za strukturu spremite u datoteku **`vlakovi.h`**, a implementaciju u datoteku **`vlakovi.cpp`**.

Struktura **`vlakovi`**

```
struct vlakovi{

map<pair<string, string>, int> cjenik;

void dodaj(string a, string b, int c);
void ukloni(string a, string b);
list<string> susjednaMjesta(string s);
pair<string,string> najskupljaLinija();
set<string> mjesta();
vector<pair<string, string> > linijeJeftinijeOd(int c);
bool mozeJeftinije(string a, string b);

};
```

`vlakovi` – Funkcije članice

- **`void dodaj(string a, string b, int cij)`**

  Dodaje **cijenu** *prevoza od mjesta* **a** *do mjesta* **b** *(i od* **b** *do* **a***). Ako već postoji vlak koji povezuje ta mjesta, promijeni staru cijenu u vrijednost* **cij.**

- **`void ukloni(string a, string b)`**

  Briše željezničku liniju koja povezuje mjesta **a** *i* **b.**

  - **`list<string> susjednaMjesta(string s)`**

  Vraća listu sa popisom imena koja su sa mjestom **s** *povezana direktnom linijom. Redoslijed mora odgovarati redoslijedu pojavljivanja u mapi.*

  - **`pair<string, string> najskupljaLinija();`**

  Vraća dirketnu liniju sa najvišom cijenom. Ukoliko ima više direktnih linija sa najvišom cijenom, vraća onu koja se nalazi zadnja u mapi.

- **set<string> mjesta()**

Vraća skup mjesta koja su povezana nekom željezničkom linijom u mapi. Redoslijed mora odgovarati redoslijedu pojavljivanja u mapi.

- **`vector<pair<string, string> > linijeJeftinijeOd(int ogr)`**

  Vraća vektor koji sadrži (direktne) linije iz željezničke mreže sa cijenom karte manjom od **ogr** *(ali BEZ cijene, samo imena krajnjih stanica). Redoslijed podataka u vektoru mora odgovarati redoslijedu kojeg su odgovarajuće linije imale u mapi.*

- **`bool mozeJeftinije(string a, string b)`**

  Vraća **true** *ako postoji niz mjesta* **x1=a, x2, x3,.., xk=b,** *takvih da su svaka dva uzastopna povezana direktnom linijom i* **cjenik(x1,x2)+cjenik(x2,x3)+...+cjenik(x(k-1),xk)<cjenik(a,b) .** *Također vraća* **true** *ako* **a** *i* **b** *nisu direktno povezane, ali postoji put sa presjedanjem. Ako ne postoji povoljniji indirektni put, vraća* **false.**

Primjer klijentskog programa

```
#include <map>
#include <utility>
#include <list>
#include "vlakovi.h"
#include <iostream>
#include <string>
#include <set>
#include <vector>
using namespace std;

void ispisMape(map<pair<string, string>, int> p){
    map<pair<string, string>, int>::iterator i;
    for(i=p.begin();i!=p.end();i++) cout << i->first.first<<" - "<<i->first.second<<"  "<< i->second<<"     ";
    cout<<endl;
}

void ispisListe(list<string> p){
    for(list<string>::iterator i=p.begin();i!=p.end();i++) cout << *i <<"     ";
    cout<<endl;
}
void ispisSeta(set<string> p){
    for(set<string>::iterator i=p.begin();i!=p.end();i++) cout << *i <<"     ";
    cout<<endl;
}

void ispisVektora(vector<pair<string, string> > p){
    for(int i=0;i<p.size();i++) cout << (p[i]).first<<"  "<<(p[i]).second<<"     ";
    cout<<endl;
}

int main(void)
{
    vlakovi test;
    test.dodaj("split","perkovic",30); test.dodaj("split","perkovic",40); test.dodaj("zagreb","karlovac",60);

test.dodaj("zagreb","rijeka",110); test.dodaj("zagreb","vinkovci",160); test.dodaj("karlovac","rijeka",45);
    test.dodaj("split","gospic",120); test.dodaj("gospic","karlovac",40);
    //ispisMape(test.cjenik);
    //    gospic-karlovac 40  gospic-split 120  karlovac-gospic 40 karlovac-rijeka 45 karlovac-zagreb 60
perkovic-split 40  rijeka-karlovac 45  rijeka-zagreb 110  split-gospic 120 split-perkovic 40  vinkovci-zagreb
160  zagreb-karlovac 60  zagreb-rijeka 110  zagreb-vinkovci 160

    test.ukloni("zagreb", "vinkovci");
    //ispisMape(test.cjenik);
```

```
    //    gospic-karlovac 40  gospic-split 120  karlovac-gospic 40 karlovac-rijeka 45 karlovac-zagreb 60
perkovic-split 40  rijeka-karlovac 45  rijeka-zagreb 110  split-gospic 120 split-perkovic 40  zagreb-karlovac
60  zagreb-rijeka 110

    ispisListe(test.susjednaMjesta("zagreb"));
    //    karlovac rijeka

    cout<<test.najskupljaLinija().first<<" "<<test.najskupljaLinija().second<<endl;
    //    split gospic

    ispisSeta(test.mjesta());
    //    gospic     karlovac     perkovic     rijeka     split     zagreb

    ispisVektora(test.linijeJeftinijeOd(41));
    //    gospic karlovac     karlovac gospic     perkovic split     split perkovic

    cout<<test.mozeJeftinije("zagreb", "rijeka")<<endl;
    //    1

    return 0;
}
```

Opće napomene

- Struktura, funkcije i datoteke koje šaljete moraju se zvati točno onako kako je zadano u zadatku. Pazite na mala i velika slova!
- Trebate poslati samo sučelje i implementaciju. U datotekama koje šaljete ne smije se nalaziti funkcija main()!
- nijedna funkcija ne smije ništa učitavati s tipkovnice ili neke datoteke, niti išta ispisivati na ekran ili u neku datoteku.
- Svaki od main-ova pomoću kojih testiramo ispravnost vašeg programa neće pozivati sve gore navedene funkcije. Stoga, ako neku od funkcija ne znate napisati ipak možete dobiti koji bod (u tom slučaju tu funkciju nemojte navesti niti u .h niti u .cpp datoteci ili napravite neku trivijalnu implementaciju).

Ispravnost implementacija koje napišete bit će provjerena tako da ćemo mi napisati razne klijentske programe koji će deklarirati nekoliko varijabli zadane strukture, i na njima pozivati funkcije koje ste trebali napisati. Ako se poslani programi ne budu uspješno povezivali (linkali) s našim klijentskim programima, smatrat će se neispravnima.

Neki klijentski programi provjeravat će samo neke jednostavnije funkcije, dok će neki provjeravati sve funkcije koje trebate napisati. Provjera je potpuno automatska, tako da je od presudne važnosti da se pridržavate specifikacije. Nepridržavanje lako može uzrokovati osvojenih 0 bodova iz zadaće!

Naravno, za provjeru radi li implementacija prije nego što je pošaljete, preporučuje se da je testirate pomoću nekog klijentskog programa. No taj klijentski program ne šaljete!
Za sva pitanja vezana uz ovu zadaću javite se asistentu Poliću na frclopir+rp1@gmail.com.