Zadatak:

Napišite sučelje i implementaciju za strukturu **`graf`** koja opisuje usmjereni graf. Struktura **`graf`** sadrži mapu koja dvama vrhovima pridružuje pozitivnu cjelobrojnu težinu usmjerenog brida. Sučelje za strukturu spremite u datoteku **`graf.h`**, a implementaciju u datoteku **`graf.cpp`**.

Struktura **`graf`**

```
struct graf {
  map<pair<string, string>, int> bridovi;

  void dodaj(string p, string d, int n);
  void ukloni(string p, string d);
  int stupanj1(string p);
  int stupanj2(string p);
  list<string> vrhovi();
  pair<string, string> najlaksiBridTeziOd(int c);
  set<string> dohvatljiviIz(string s);

};
```

`graf`– Funkcije članice

- **`void dodaj(string p, string d, int n)`**

  Dodaje brid **(p, d)** *težine* **n** *u mapu (NE i od* **d** *do* **p,** *nema simetrije). Ako već postoji brid* **(p, d)** *ili ako je* **p==d,** *ne dodaje ništa.*

- **`void ukloni(string p, string d)`**

  Briše brid **(p, d).**

  - **`int stupanj1(string vrh)`**

  Vraća broj bridova sa **`vrh`** *kao polaznom točkom.*

  - **`int stupanj2(string vrh)`**

  Vraća broj bridova sa **`vrh`** *kao završnom točkom.*

- **list<string> vrhovi()**

  Vraća listu vrhova u grafu. Poredak treba odgovarati poretku kojim se vrhovi "nalaze" u mapi. Drugim riječima, ako se prvi brid koji sadrži vrh **a** *nalazi prije prvog brida koji sadrži* **b,** *onda* **a** *treba biti prije* **b** *u listi. Ako se* **a** *i* **b** *prvi put u mapi pojavljuju u istom bridu, polazni vrh treba biti prije u listi.*

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

Vraća par koji predstavlja brid sa najmanjom težinom u grafu među svim bridovima težine veće od **ogr.** *Ako postoji više takvih parova, treba vratiti onaj koji se nalazi prije u mapi.*

- **`set<string> dohvatljiviIz(string a)`**

  Vraća skup stringova koji predstavljaju sve vrhove do kojih postoji put iz vrha **a,** *uključno sa vrhom argumentom funkcije***.** *Poredak podataka u setu nije bitan.*

Primjer klijentskog programa

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

void ispisMape(map<pair<string, string>, int> p){
    for(map<pair<string, string>, int>::iterator 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 ispisSkupa(set<string> p){
    for(set<string>::iterator i=p.begin();i!=p.end();i++) cout << *i <<" ";
    cout<<endl;
}
int main(void)
{
    graf test;
    test.dodaj("a","b",30); test.dodaj("a","c",40); test.dodaj("a","g",60);
    test.dodaj("b","d",110); test.dodaj("a","d",160); test.dodaj("f","g",45);
    test.dodaj("g","g",120); test.dodaj("g","a",40);test.dodaj("g","a",40);
    test.dodaj("a","b",40);test.dodaj("c","a",40);test.dodaj("e","a",40);
    //ispisMape(test.bridovi);
    //    a - b 30     a - c 40     a - d 160     a - g 60     b - d 110     c - a 40     e - a 40     f - g
45     g - a 40


    test.ukloni("f", "g");
    //ispisMape(test.bridovi);
    //    a - b 30     a - c 40     a - d 160     a - g 60     b - d 110     c - a 40     e - a 40     g - a
40


    cout<<test.stupanj1("a")<<"      "<<test.stupanj2("d")<<endl;
    //    4      2


    ispisListe(test.vrhovi());
    //    a b c d e g


    cout<<test.najlaksiBridTeziOd(50).first<<" "<<test.najlaksiBridTeziOd(50).second<<endl;
    //    a g


    ispisSkupa(test.dohvatljiviIz("a"));
    //    a b c d g


    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.