Zadatak

Napišite sučelje i implementaciju za strukturu **maskare**. Struktura **maskare** sadrži listu kuća koja se nalaze u ulici. Svaka kuća je jedinstveno određeno imenom vlasnika i taj vlasnik za maškare djeci uzima novac ili daje novac. Kuće su u ulici poredane onim redoslijedom kojim ih dodajemo. Sučelje za strukturu spremite u datoteku **maskare.h**, a implementaciju u datoteku **maskare.cpp**.

Struktura maskare

```
struct maskare
{
    list<pair<string, int> > kuce;

    list<pair<string, int> > vratiListu(void) {return kuce;}
    void dodaj(string ime, int novac);
    void izbrisi(string ime);
    string izbjegavatIme(void);
    int ukupnoKuna(string pocetak, string kraj, int pocetniIznos);
    map<string, int> dobriJudi(void);
    set<string> pronadjiImena(string substr, int novac);
    pair<string, string> pametniObilazak(int pocetniIznos);
};
```

Funkcije članice

- **void dodaj(string ime, int novac)**
  Dodaje novu kuću na kraj liste **kuce**. **Ime** *je ime vlasnika kuće,* a **novac** *je iznos koji vlasnik daje/krade djeci za maškare. Ako je* **novac** *pozitivan onda vlasnik daje novac, a ako je negativan onda ga uzima. Ako dijete ima manje novca kod sebe nego što mu vlasnik želi uzeti onda mu vlasnik uzima sve što ima, tj. djetetu ostaje nula kuna.*

- **void izbrisi(string ime)**
  Izbacuje kuću danog vlasnika iz liste. Ostale kuće ostaju poredane prema redoslijedu kojeg su imali i prije izvršavanja funkcije.

- **string izbjegavatIme(void);**
  Vraća vlasnika koji djeci uzima najviše novaca. Ako svi vlasnici daju novce tada vraća onog koji najmanje daje. Ako postoji više takvih vlasnika vraća onog koji živi u kući koja je najbliža početku ulice (tj. u listi se nalazi najbliže početku liste).

- **int ukupnoKuna(string pocetak, string kraj, int pocetniIznos)**
  Vraća ukupan broj kuna koje dijete ima nakon obilaska kuća. Dijete kreće od kuće čiji vlasnik ima ime **pocetak** *i redom obilazi sve kuće završno s kućom čiji vlasnik ima ime* **kraj**. *Dijete prije obilaska ima* **pocetnilznos** *kuna kod sebe.*

- **map<string, int> dobriJudi(void)**
  Vraća map<string, int> *koja se sastoji od svih vlasnika koji djeci daju novac (strogo veće od nula). Ključ je ime vlasnika, a vrijednost je iznos koji vlasnik daje.*

- **set<string> pronadjiImena(string substr, int novac)**
  Vraća skup imena vlasnika koji daju iznos strogo veći od **novac** *(uvijek veći od nula) te njihova imena sadrže podriječ* **substr**.

- **pair<string, string> pametniObilazak(int pocetniIznos)**

  Vraća par imena vlasnika takav da je krajnji iznos koji dijete skupi obilazeći redom sve kuće krenuvši od prvog vlasnika (first u vraćenom vraćenom paru) do drugog vlasnika (second u vraćenom paru) je maksimalan. Prvi vlasnik se uvijek nalazi prije (ili je jednak) drugog vlasnika u poretku kuća. Ako postoj više takvih parova uzimamo onaj kod kojeg se kuća prvog vlasnika nalazi bliže početku liste, a ako ima i takvih više onda se vraća par kod kojeg je broj kuća koje se trebaju obići najmanji. Dijete na početku ima **pocetnilznos** *kuna kod sebe. Možete pretpostaviti da uvijek postoji jedan vlasnik koji daje novac.*

Primjer klijentskog programa

```
#include <iostream>
#include <map>
#include <set>
#include <list>
#include <string>
#include <utility>
#include "maskare.h"

void ispisiListu(list<pair<string, int> > L)
{
    list<pair<string, int> >::iterator it;

    for(it = L.begin(); it != L.end(); it++)
        cout << it->first << " " << it->second << " ";
}

void ispisiSkup(set<string> S)
{
    set<string>::iterator it;

    for(it = S.begin(); it != S.end(); it++)
        cout << *it << " ";
}

void ispisiMap(map<string, int> M)
{
    map<string, int>::iterator it;

    for(it = M.begin(); it != M.end(); it++)
        cout << it->first << " " << it->second << " ";
}


int main(void)
{
    maskare mask;

    mask.dodaj("ante", 5); mask.dodaj("josip", -2); mask.dodaj("bosko", 3);
    mask.dodaj("ana", -4); mask.dodaj("ivana", 1); mask.dodaj("marko", 2);
    mask.dodaj("mate", 3); mask.dodaj("jelena", -3); mask.dodaj("filip", 5);
    mask.dodaj("petar", -1); mask.dodaj("karlo",1);mask.dodaj("maja", -10);
    mask.dodaj("igor", -1); mask.dodaj("tonci", 2); mask.dodaj("kristina", -1);
    mask.dodaj("alma", 7); mask.izbrisi("bosko");

    ispisiListu(mask.vratiListu());
    //    ante 5 josip -2 ana -4 ivana 1 marko 2 mate 3 jelena -3 filip 5 petar -1 karlo 1 maja -10 igor -1
tonci 2 kristina -1 alma 7

    cout << endl;
    cout << mask.izbjegavatIme() << endl;
    //    maja

    cout << mask.ukupnoKuna("josip", "marko", 20) << endl;
    //    17

    cout << mask.ukupnoKuna("josip", "marko", 1) << endl;
    //    3


    ispisiMap(mask.dobriJudi());
    //    alma 7 ante 5 filip 5 ivana 1 karlo 1 marko 2 mate 3 tonci 2

    cout << endl;
    ispisiSkup(mask.pronadjiImena("ma", 2));
    //    alma mate

    cout << endl;
    pair<string, string> p = mask.pametniObilazak(100);
    cout << p.first << " " << p.second << endl;
    //    ivana filip

    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 Vujčiću na jvujcic+rp1@gmail.com.