Zadatak

Napišite sučelje i implementaciju za strukturu **igra**. Struktura **igra** sadrži listu polja. Svako polje je jedinstveno određeno svojim identifikacijskim brojem. Također, svako polje sadrži broj bodova(mogu biti pozitivni i negativni). Polja su poredana onim redoslijedom kojim ih dodajemo. Igru igraju dva igrača. Prvom igraču pripadaju polja s parnim identifikacijskim brojem, a drugom igraču neparna. Svaki igrač na početku odabere dva polja (mogu biti ista). Pobjeđuje onaj čija je suma bodova njihovih polja koja se nalaze između dva odabrana polja (uključujući i ta dva polja) veća. Sučelje za strukturu spremite u datoteku **igra.h**, a implementaciju u datoteku **igra.cpp**.

Struktura igra

```
struct igra
{
    list<pair<int, int> > listaPolja;

    list<pair<int, int> > vratiListu(void) {return listaPolja;}
    void dodajPolje(int ID, int bodovi);
    void izbrisiPolje(int ID);
    vector<pair<int, int> > polja(int igrac);
    map<int, int> viseOd(int bodovi);
    int pobjednik(int poc1, int kraj1, int poc2, int kraj2);
    pair<int, int> optimalnaIgra(int igrac);
};
```

Funkcije članice

- **void dodajPolje(int ID, int bodovi)**
  Dodaje novo polje na kraj liste **listaPolja**. *Identifikacijski broj polja je* **ID** *(zapisuje se u prvi član od* pair<int, int>), a **bodovi** *označuju koliko bodova nosi to polje (zapisuje se u drugi član od pair<int, int>).*

- **void izbrisiPolje(int ID)**
  Izbacuje traženo polje iz liste. Ostala polja ostaju poredana prema redoslijedu kojeg su imali i prije izvršavanja funkcije.

- **vector<pair<int, int> > polja(int igrac)**
  Vraća vector polja koja pripadaju igraču **igrac** *(jednako* 1 *ili* 2, što odgovara prvom ili drugom igraču). *Polja u vectoru imaju isti redoslijed kao u originalnoj listi.*

- **map<int, int> viseOd(int bodovi)**
  Vraća map<int, int> koje se sastoji od svih polja koji nose strogo više od **bodovi**. *Ključ je identifikacijski broj polja, a vrijednost je broj bodova.*

- **int pobjednik(int poc1, int kraj1, int poc2, int kraj2)**
  Prvi igrač izabere polja s identifikacijom **poc1** *i* **kraj1**, *dok drugi igrač izabere polja s identifikacijom* **poc2** *i* **kraj2** *(***poc1** *se nalazi uvijek prije ili je jednako* **kraj1**, *i* **poc2** *se nalazi uvijek prije ili je jednako* **kraj2**). *Funkcija vraća* 1 *ako je prvi igrač pobijedio,* 2 *ako je drugi igrač pobijedio i* 0 *ako je izjednačeno.*

- **pair<int, int> optimalnaIgra(int igrac)**
  Vraća par identifikacijski brojeva polja za koje igrač **igrac** *(jednako* 1 *ili* 2, *što odgovara prvom i drugom igraču) ima optimalnu igru (skupi maksimalni broj bodova). Prvi broj odgovara početnom polju (first u vraćenom paru), a drugi krajnjem polju (second u vraćenom paru) koje igrač bira. Ako postoji više takvih parova tada uzimamo onog kod kojeg se prvo polje nalazi bliže početku liste, a ako ima i takvih više tada uzimamo onaj par koji sadrži najmanje polja između. Možete pretpostaviti da će za svakog igrača postojati makar jedno polje s pozitivnim brojem bodova.*

Primjer klijentskog programa

```
#include "igra.h"
#include <iostream>
#include <map>
#include <vector>
#include <list>
#include <utility>

using namespace std;

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

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

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

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

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


int main(void)
{
    igra game;

    game.dodajPolje(112, -4); game.dodajPolje(226, 5); game.dodajPolje(327, 2);
    game.dodajPolje(158, -3); game.dodajPolje(445, 5); game.dodajPolje(649, 7);
    game.dodajPolje(654, 4); game.dodajPolje(120, -2); game.dodajPolje(821, -4);
    game.dodajPolje(415, -1); game.dodajPolje(117, 4); game.dodajPolje(926, 1);
    game.dodajPolje(258, -15); game.dodajPolje(202, 6); game.izbrisiPolje(445);

    ispisiListu(game.vratiListu());
    //   112 -4 226 5 327 2 158 -3 649 7 654 4 120 -2 821 -4 415 -1 117 4 926 1 258 -15 202 6

    cout << endl;
    ispisiVector(game.polja(1));
    //   112 -4 226 5 158 -3 654 4 120 -2 926 1 258 -15 202 6

    cout << endl;
    ispisiVector(game.polja(2));
    //   327 2 649 7 821 -4 415 -1 117 4

    cout << endl;
    ispisiMap(game.viseOd(2));
    //    117 4 202 6 226 5 649 7 654 4

    cout << endl;
    cout << game.pobjednik(226, 120, 649, 117) << endl;
    //    2

    pair<int, int> p = game.optimalnaIgra(1);
    cout << p.first << " " << p.second << endl;
    //    226 654

    p = game.optimalnaIgra(2);
    cout << p.first << " " << p.second << endl;
    //    327 649

    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.