Zadatak 1 - Duration tip podatka

Potrebno je implementirati `c++` tip podatka `Duration` koji korisniku omogućuje pohranu proteklog vremena. Takav podatak sadrži potrebne konstruktora, mogućnosti aritmetičkih, relacionih i logičkih operatora, te korisniku može dati informaciju o tome koliko je proteklo sekundi, minuta i sati.

Pored default konstruktora, koji inicijalizira vrijednost na 0, postoji konstruktor koji prihvata broj proteklih sekundi kao argument. Ovaj konstruktor prima i vrijednosti veće od 60, što će imati značenje da je prošlo više od jedne minute. Nekoliko primjera korištenja možete vidjeti u kodu ispod:

```
Duration d;
d.get_s() \\ vraća 0
d.get_m() \\ vraća 0
d.get_h() \\ vraća 0

Duration d{30};
d.get_s() \\ vraća 30
d.get_m() \\ vraća 0
d.get_h() \\ vraća 0

Duration d{125};
d.get_s() \\ vraća 5
d.get_m() \\ vraća 2
d.get_h() \\ vraća 0

Duration d{3600};
d.get_s() \\ vraća 0
d.get_m() \\ vraća 0
d.get_h() \\ vraća 1
```

Drugi tip konstuktora prihvata vrijednosti sati, minuta i sekundi. Potrebno je da ovaj konstruktor baci iznimku tipa `std::out_of_range` ako se neka od vrijednosti nađe izvan dozvoljenog opsega.

```
Duration d{30, 20, 10};
d.get_s() \\ vraća 10
d.get_m() \\ vraća 20
d.get_h() \\ vraća 30

Duration d{10, 20, 100}; // ova linija baca iznimku zbog inicijalizacije sekundi
```

Pored `get` metoda, koje ste vidjeli u primjerima iznad, ovaj tip podatka podržava i `set` metode. Oni također bacaju iznimku, kao i prethodni konstruktor. `Set`ere je moguže pozivati posebno, ali i lančano kao u primjerima ispod:

```
Duration d;
d.set_s(56);
d.get_s(); // vraća 56
```

```
d.set_s(12).set_h(30).set_m(45); // Vrijednost se postavljaju na 30h 45m 12s

d.set_m(60); // baca iznimku tipa std::out_of_range
```

Relacione operacije obuhvataju operatore `==, !=, <, >, <= i >=`. Ovaj tip se može porediti samo sa drugim objektom tipa `Duration`.

```
Duration d1{10, 35, 15};
Duration d2{10, 40, 15};
d1 == d1; // true;
d1 != d1; // false;
d1 > d2; // true
d1 < d2; // false
d1 >= d2; // true
d1 <= d2; // false
```

Da bi osposobili logičke operacije, potrebno je implementirati dva unarna operatora. Prvi je operator cast (pretvaranje) u bool, a drugi je unarni operator!. Objekat tipa `Duration` se evaluira na vrijednost tačno ako je u njemu pohranjena neka nenulta vrijednost. Da bi implementacija bila jednostavnija, potpisi ovih metoda su dati ispod:

```
class Duration {

    // prvi dio klase

    explicit operator bool() const {
        // vas kod ovdje
    }
    bool operator!() const {
        // vas kod ovdje
    }

    // drugi dio klase

}
```

Implementacijom ovih operatora, postaje moguće pisati kod kao u nastavku:

```
Duration d;
if (d) { ... }
if (!d) { ... }
```

Aritmetičke operacije uključuju operatore `+=, +, -=, -, *=, *, /= i /`. Operacije sabiranja i oduzimanja moguće su sa drugim objektom tipa `Duration`, dok se operacije množenja i dijeljenja moguće sa varijablom tipa `int`. Oduzimanje većeg `Duration` tipa od manjeg baca iznimku tipa `std::out_of_range`.

```
Duration d1{30};
Duration d2{1,30,0};
```

```
d1 = d1 + d2; // d1 = 01:30:30
d1 += d1;     // d1 = 03:01:00
d1 = d1 - d2; // d1 = 01:31:00

d2 = d1 + d1;
d1 - d2;            // ova operacija bi bacila iznimku jer je d2 > d1
```

```
Duration d1{3,10,10};
d1 = d1 * 3;    // d1 = 09:30:30
d1 /= 2;    // d1 = 04:45:15
```

Ispis vrijedonsti sa `std::ostream` objektom i `operator<<` pravi format `hh:mm:ss`. Taj format se očekuje i pri čitanju `Duration` objekta sa standardnog ulaza.

```
Duration d{1, 30, 5};
std::cout << d << std::endl;  // ispis na ekran 01:30:05
std::cin >> d;  // korisnik unosi 10:05:00
d.get_s()   // vraća 0
d.get_m()   // vraća 5
d.get_h()   // vraća 10
```

Dodatno, potrebno je napraviti konstruktor koji prihvata `std::string` formatiran kao `hh:mm:ss`

```
Duration d{"10:30:25"};
d.get_s(); // vraca 25
d.get_m(); // vraca 30
d.get_h(); // vraca 10
```

Testiranje Vašeg rješenja

U sklopu zadatka imate file-ove koji će testirati Vašu implementaciju. O svakom testnom fileu možete razmišljati kao o skupini manjih main funkcija, gdje svaki main testira jedan dio jedan dio Vašeg rješenja.

Za pokretanje testova potrebno je pokrenuti skriptu `./run_tests.sh` koja će kompajlirati i pokretati redom, jedan po jedan, 14 test fileova.

- test_01.cpp - testira default konstruktor, konstruktor koji prima sekunde i get metode,
- test_02.cpp - testira konstruktor koji prima 3 vrijednosti, te copy i move konstruktor i operator=,
- test_03.cpp - testira set metode,
- test_04.cpp - testira operatore `==, !=, >, >=, <, <=`,
- test_05.cpp - testira operatore bool i !,
- test_06.cpp - testira operatore +=,
- test_07.cpp - testira operatore +,
- test_08.cpp - testira operatore -=,

- test_09.cpp - testira operatore -,
- test_10.cpp - testira operatore $= i$,
- test_11.cpp - testira operatore /= i /,
- test_12.cpp - testira operator«,
- test_13.cpp - testira operator»,
- test_14.cpp - testira konstruktore koji primaju string.

Pri obradi jednog test filea, 3 stvari se mogu dogoditi: - File ne može da se kompajlira. Moguće da niste implementirali neku metodu koju taj file testira, potpis metode nije odgovarajući ili imate druge sintaksne greške, - file se kompajlira i pokrene, ali Vam ispiše `ERROR` poruke. Imate bug u svom rješenju. Možete otvoriti testni file i pogledati koji se metodi pozivaju. Moguće je da ste pogriješili u samo nekom specijalnom slučaju. - file se kompajlira i pokrene, te dobijete poruku da su svi testovi prošli bez grešaka.

Zadatak 2 - Rational tip podatka

Potrebno je implementirati Rational tip podatka koji simulira rad sa racionalnim brojevima (razlomcima). Deklaracije svih metoda, operatora i konstruktora su dati u fajlu `Rational.hpp`. Neophodno je učitati ulazne podatke iz fajla input.txt. U fajlu su dati sljedeći podaci:

```
2/3
4/6
5/1
10/0
5/1
1/4
7/7
12/4
15/3
24/8
4
23#2
23a
ab/cd
```

Pri čemu, zadnja tri reda u fajlu predstavljaju nevalidan ulaz kojeg treba na odgovarajući način interpretirati. Također, nazivnik nikada ne smije biti 0. Ukoliko se desi bilo kakva greška, baciti iznimku tipa `std::invalid_argument`. Prilikom ispisa razlomka na standardni izlaz neophodno je razlomak skratiti sa najvećim zajedničkim sadržiocem (gcd). U tu svrhu možemo iskoristi funkciju koja prima brojnik i nazivnik te vraća najveći zajednički djelilac kojeg možemo iskoristiti za kraćenje razlomka.

```
int gcd(int numerator, int denominator) {
  const int num = numerator;
```

```
  const int denom = denominator;
  int shift;
  for (shift = 0; ((numerator | denominator) & 1) == 0; ++shift) {
    numerator >>= 1;
    denominator >>= 1;
  }
  while ((numerator & 1) == 0) {
    numerator >>= 1;
  }
  while (denominator != 0) {
    while ((denominator & 1) == 0) {
      denominator >>= 1;
    }
    if (numerator > denominator) {
      std::swap(numerator, denominator);
    }
    denominator -= numerator;
  }

  return numerator << shift;
}
```

Dakle, ispis nakon normalizacije razlomka treba da bude:

```
2/3
2/3 -> Normalizacija sa 2
5 -> Nazivnik 1 nije potrebno ispisivatij
1/4
1 -> 7/7 Normalizacija sa 7
3 -> 12/4 Normalizacija sa 4
5 -> 15/3 Normalizacija sa 3
3 -> 24/8 Normalizacija sa 8
4
```

Opis metoda:

`Rational();`

Default konstruktor treba da postavi vrijednosti za brojnik 0, za nazivnik 1 (Nazivnik nikad ne smije biti 0!)

```
  Rational(int numerator, int denominator);
```

Konstruktor parametriziran sa dva cijela broja. Pri čemu treba voditi računa da `denominator` ne smije biti 0! U slučaju da je nazivnik jednak 0, generisati iznimku tipa `std::invalid_argument`.

```
  Rational(const char* rational);
  Rational(const std::string&);
```

Oba konstruktora imaju slično (identično ponašanje). Neophodno je niz karaktera ili string parsirati na odgovarajući način kako bi dohvatili vrijednosti za brojnik i nazivnik. Format koji je validan: “2/3”, “3”, “1/5”, i slično. Međutim ne smije se dozvoliti ne validan format kao: delimiter različit od /, izraz sadržava nešto što nije cifra... Za dohvatanje vrijednosti možete iskoristiti klasu `istringstream`. To je jedan od prijedloga, ali bilo koje rješenje je prihvatljivo!

```
Rational(const Rational& rational);
Rational(Rational&& rational);
```

Copy i move konstruktori. Prilikom implementacije move konstruktora, objekat koji se move-a treba vratiti u default stanje (brojnik 0, nazivnik 1).

```
Rational& operator=(const Rational& rational);
Rational& operator=(Rational&& rational);
```

Copy i move operator=. Prilikom implementacije move operatora=, objekat koji se move-a treba vratiti u default stanje (brojnik 0, nazivnik 1).

```
Rational operator+(const Rational& rational) const;
Rational operator-(const Rational& rational) const;
```

Prilikom implementacije operatora+ i operatora-, neophodno je primijeniti pravila za sabiranje dva razlomka. Oba razlomka treba svesti na zajednički sadržilac te adekvatno skratiti razlomke sa najvećim zajedničkim djeliocem. Za pronalazak najvećeg zajedničkog djelioca možemo iskoristiti funkciju `gcd` koja je data u opisu zadatka.

```
Rational operator+(int numerator) const;
Rational operator-(int numerator) const;
```

Operatori koji omogućavaja sabiranje i oduzimanje razlomaka sa cijelim brojem (nazivnik jednak 1). Kao na primjer, 3/4 + 3, rezultat sabiranja treba da bude 15/4 jer se 3 može svesti na najmanji zajednički sadržilac kao 12/4.

```
Rational operator*(const Rational& rational) const;
Rational operator*(int numerator) const;
Rational operator/(const Rational& rational) const;
Rational operator/(int numerator) const;
```

Operatori množenja i dijeljenja za razlomke trebaju da poštuju pravilan način množenja i dijeljenja razlomaka. Nakon završene operacije razlomak je neophodno normalizirati (skratiti sa najvećim zajedničkim djeliocem).

```
Rational operator^(int) const;
```

Operator ^ implementira operaciju stepenovanja razlomka sa cijelim brojem (Voditi računa o normalizaciji razlomaka).

```
Rational& operator++();
Rational operator++(int);
```

```
Rational& operator--();
Rational operator--(int);
```

Prefix i postfix operatori ++ i – odgovaraju operaciji sabiranja/oduzimanja razlomka sa 1 (1/1).

```
bool operator==(const Rational& rational) const;
bool operator==(const char* rational) const;
bool operator!=(const Rational& rational) const;
bool operator!=(const char* rational) const;
```

Logički operatori koji porede dva razlomka. Razlomci su jednaki ako su im brojnici i nazivnici jednaki. Pri čemu voditi računa da su 3/4 i 6/8 jednaki razlomci jer se drugi razlomak može normalizirati na 3/4.

```
const int numerator() const;
const int denominator() const;
```

Getteri koji dohvataju vrijednosti brojnika i nazivnika.

Napomena: Neophodno je implementirati sve navedene funkcionalnosti racionalnog broja u fajlu Rational.cpp. Učitavanje ulaznih podataka (input.txt) odraditi u fajlu main.cpp. Neophodno je nakon završenog učitavanja testirati sve operatore te provjeriti njihovu ispravnost! Također, neophodno je tretirati sve moguće iznimke koje se mogu pojaviti prilikom izvršavanje neke operacije. Program ni u jednom slučaju ne smije terminirati zbog neuhvaćene iznimke!

Zadatak 3 - F1 timovi

Potrebno je ucitati podatke iz fajla u odgovarajuce tipove podataka unutar programa, odraditi transformaciju podataka na nacin objasnjen u nastavku, te je potrebno sortirane podatke ispisati u novi fajl. Format novog fajla je isti kao u prvobitnom fajlu.

Potrebno je izvrsiti sortiranje podataka po odredjenom kriteriju i filtrirati timove na osnovu zadatog uslova.

- Ucitavanje podataka: Ucitati podatke iz CSV fajla *f1_teams.csv* koji se nalazi u prilogu.
- Sortiranje podataka: Sortirati timove po broju osvojenih prvenstava (number_of_championships) u opadajucem redoslijedu.
- Filtriranje podataka: Ukloniti timove koji nemaju nijedno osvojeno prvenstvo.
- Spremanje rezultata: Sacuvati rezultirajuce podatke u novi CSV fajl pod nazivom *f1_teams_sorted.csv*.
- Objediniti podatke: Ako se u fileu nalazi vise istih timova (isto ime i mjesto osnivanja) spojiti te unose u jedan. Spojiti ih na nacin da se za godinu osnivanja uzima najniza vrijednost, a budzet i broj osvojenih prvenstava akumulira, npr:

```
Ferrari,Italy,1992,0,20
Ferrari,Italy,1985,4,30
```

se trebaju grupisati u:

```
Ferrari,Italy,1985,4,50
```

Problem rijesiti koristeci C++ file API prateci OOP koncepte.