Zadatak 1 - MojVektor kontejner

Potrebno je implementirati generički MojVektor kontejner koji ima slične funkcionalnosti kao std::vector. Kontejner je implementiran kao lista nizom s tim da je proširen sa mogućnošću korištenja bidirekcionih iteratora. Deklaracija svih metoda je data u fajlu MojVektor.hpp. Neophodno je implementirati sve metode kako bi svi testovi prošli. Ispravna implementacija je neophodna s obzirom da se ostali zadaci koriste implementaciju ovog kontejnera.

Opis metoda:

```
MojVektor();
```

Default konstruktor treba da alocira prostor za 10 elemenata.

```
MojVektor(const std::initializer_list<T>&);
```

Konstruktor parametriziran listom za inicijalizaciju. Novoalocirani kontejner treba imati kapacitet koji je jednak veličini ove liste. Za kopiranje elemenata koristiti std::copy algoritam umjesto kopiranja korištenjem petlji.

```
MojVektor(const MojVektor&);
MojVektor& operator=(MojVektor&&);
```

Copy konstruktor i copy operator=

```
MojVektor(MojVektor&&);
MojVektor& operator=(MojVektor&&);
```

Move konstruktor i move operator=

```
~MojVektor();
```

MojVektor destruktor

```
MojVektor& push_back(const T&);
```

Metod push_back dodaje novi element na kraj kontejnera, pri čemu treba voditi računa o kapacitetu kontejnera. Ako je kapacitet popunjen, izvršiti realokaciju te kapacitet proširiti za 2 puta.

```
MojVektor& push_front(const T&);
```

Metod push_back dodaje novi element na početak kontejnera, pri čemu treba voditi računa o kapacitetu kontejnera. Ako je kapacitet popunjen, izvršiti realokaciju te kapacitet proširiti za 2 puta.

```
size_t size() const;
```

Metod size vraća ukupan broj elemenata u kontejneru.

```
T& at(size_t index) const;
```

Metod at dohvata element na poziciji index. Ovaj metod baca iznimku tipa std::out_of_range u slučaju pristupa elementima van opsega.

```
T& operator[](size_t) const;
```

Operator [] dohvata element na poziciji index. Ovaj metod ne baca nikakvu iznimku (identično ponašanje kao kod std::vector kontejnera)

```
void clear();
```

Metod clear briše sve elemente iz kontejnera.

```
void resize(size_t newSize);
```

Metod resize postavlja novi size za kontejner pri čemu:

- Ako je newSize < size_ -> size_ postaviti na newSize (nije potrebna realokacija)
- Ako je newSize > capacity -> Proširiti kontejner u veličini newSize, size_ i capacity_ postaviti na newSize. Zadržati stare elemente, pri čemu krajnje elemente (zadnjih newSize - OldSize elemenata) postaviti na 0.

```
MojVektor& pop_back();
```

Metod pop_back uklanja zadnji element iz kontejnera, pri čemu baca iznimku tipa std::out_of_range ukoliko pokušamo ukloniti element iz praznog kontejnera.

```
MojVektor& pop_front();
```

Metod pop_front uklanja prvi element iz kontejnera, pri čemu baca iznimku tipa std::out_of_range ukoliko pokušamo ukloniti element iz praznog kontejnera.

```
MojVektor& back();
```

Metod back dohvata zadnji element iz kontejnera, pri čemu baca iznimku tipa std::out_of_range ukoliko pokušamo dohvatiti element iz praznog kontejnera.

```
MojVektor& front();
```

Metod front dohvata prvi element iz kontejnera, pri čemu baca iznimku tipa std::out_of_range ukoliko pokušamo dohvatiti element iz praznog kontejnera.

```
bool empty() const;
```

Metod empty vrši provjeru da li je kontejner prazan.

```
size_t capacity() const;
```

Metod capacity() vraća ukupan kapacitet kontejnera.

```
bool operator==(const MojVektor&) const;
bool operator!=(const MojVektor&) const;
```

Logički operatori == i != porede jednakost dva kontejnera. Dva kontejnera su jednaka ako su im veličine jednake i svi elementi na odgovarajućim pozicijama imaju istu vrijednost.

```
bool full() const;
```

Metod full() vraća true ukoliko je capacity_ jednak size_, u suprotnom vraća false.

```
MojVektor subvector(Iterator begin, Iterator end);
```

Metod subvector() kao rezultat vraća vektor čiji su elementi jednaki elementima originalnog vektora u opsegu definiranom kao [begin, end).

```
Iterator begin() const;
Iterator end() const;
```

Metod begin() vraća iterator na prvi element u vektoru, dok end() vraća iterator na jednu poziciju iza zadnjeg elementa u vektoru.

```
Iterator find(const T& value) const;
```

Metod find() vraća iterator na prvi element koji je jednak value, u suprotnom vraća end iterator.

```
Iterator erase(Iterator pos);
```

Metod erase() uklanja element sa pozicije pos te vraća iterator na element iza elementa koji je uklonjen.

```
Iterator insert(Iterator pos, const T& value);
```

Metod insert() dodaje novi element na poziciju pos.

```
Iterator rbegin() const;
```

Metod rbegin() vraća iterator na zadnji element u vektoru (reverse begin).

```
Iterator rend() const;
```

Metod rend() vraća iterator na jednu poziciju prije prvog elementa u vektoru (reverse end).

```
Iterator erase(Iterator beginIt, Iterator endIt);
```

Metod erase() uklanja sve elemente iz vektora u opsegu [beginIt, endIt) te vraća iterator na element iza zadnjeg uklonjenog elementa u opsegu.

```
void rotate();
```

Metod rotate rotira kontejner sa pocetka na kraj. Primjer: {1, 2, 3, 4, 5} -> {5, 4, 3, 2, 1}

```
void rotate(Iterator beginIt, Iterator endIt);
```

Metod rotate rotira kontejner u opsegu [beginIt, endIt). Primjer: {1, 2, 3, 4, 5} -> rotate(vec.begin() + 1, vec.begin() + 4) -> {1, 4, 3, 2, 5}

Napomena: Da bi svi testovi uspjesno prosli neophodno je implementirati sve funkcionalnosti Iterator klase.

Testovi se pokrecu komandom:

```
bash run_tests.sh
```

Ukoliko zelite pokrenuti samo odredjeni test to radite komandom:

```
bash run_tests.sh broj_testa
```

gdje je broj_testa redni broj testa, na primjer:

```
bash run_tests.sh 35
```

pokrece test 35.

Zadatak 2

Implementi program za evidenciju posuđenih knjiga u biblioteci pomoću MojVektor kontejnera kojeg ste implementirali u prethodnom zadatku.

Program treba imati sljedeće funkcionalnosti:

1. Unos nove knjige - omogućuje unos podataka o novoj knjizi (naslov, autor, izdavač, godina izdavanja, broj primjeraka).
2. Posuđivanje knjige - omogućuje posuđivanje knjige korisniku (unos podataka o korisniku, broj primjeraka koji se posuđuju).
3. Vraćanje knjige - omogućuje vraćanje posuđene knjige u biblioteku (smanjuje broj posuđenih primjeraka za broj vraćenih primjeraka).
4. Pretraga knjiga - omogućuje pretragu knjiga pomoću naslova, autora, izdavača ili godine izdavanja. Program treba čuvati podatke o knjigama i korisnicima u MojVektor kontejnerima.

Svaka knjiga se predstavlja objektom klase Book, koja ima sljedeće članove:

1. Naslov (string)
2. Autor (string)
3. Izdavač (string)
4. Godina izdavanja (int)
5. Broj primjeraka (int)

Svaki korisnik se predstavlja objektom klase User, koja ima sljedeće članove:

1. Ime (string)
2. Prezime (string)
3. ID korisnika (int)

Potrebno je implementirati metode za dodavanje, posuđivanje, vraćanje i pretragu knjiga u klasi Library. Ova klasa treba koristiti MojVektor kontejnere za čuvanje podataka o knjigama i korisnicima. Jedan korisnik može posuditi samo jedan primjerak neke knjige. Korisnik kod sebe u datom trenutku može posuditi samo jedan primjerak neke knjige.

Također, potrebno je implementirati funkciju main() koja omogućuje interakciju s korisnikom i poziva odgovarajuće metode klase Library za izvršavanje funkcionalnosti programa.

Zadatak 3

Implementi program za evidenciju posuđenih knjiga u biblioteci, kao u zadatku 2, s tim da je ovu implementaciju potrebno uraditi pomoću `std::vector` kontejnera iz standardne biblioteke.

Predaja zadace

Zadaću predajete putem google classroom-a do zadatog roka. Potrebno je predati jednu datoteku, sa imenom `ime_prezime_sp_zadaca2.tar.gz`, koja sadrži direktorij sa imenom `ime_prezime_sp_zadaca2` i poddirektorijem za svaki zadatak. U sklopu zadaće se nalaze testovi za prvi zadatak i video prezentacija kako bi trebali funkcionisati 2. i 3. zadatak.