2. kolokvij 26.06.2023. **Računarski praktikum 1**

Smijete koristiti pisane materijale ili vlastite bilješke spremljene u HOME direktoriju, web-stranicu kolegija i linkove koji vode s nje.

Ne smijete se koristiti nikakvim sredstvima komunikacije (papirići, mobitel, mail, chat, messenger i sl.). Svaki pokušaj prepisivanja i dogovaranja rezultirat će trenutnim udaljavanjem sa kolokvija. Strogo je zabranjeno fotografiranje ili pretipkavanje teksta zadataka. Iznimno, studenti koji kolokvij pišu u čitaonici, smiju upaliti hostspot na mobitelu, nakon čega mobitel moraju okrenuti licem prema stolu na prikladnoj udaljenosti. Mobitel se nakon toga više ne smije dirati do kraja izvođenja kolokvija.

Prvi zadatak (20 bodova)

Implementirajte hijerarhiju klasa: apstraktnu klasu **PO** (koja predstavlja poreznog obveznika, osobu, tvrtku ili sl), iz nje izvedenu klasu **Osoba**, te iz nje izvedenu klasu **Student**. Klasa **PO** treba imati članske varijable **oib** (**std::string**) i **postotak** (**double**, koji postotak prihoda se oporezuje). Klasa **Osoba** treba dodatno imati članske varijable **ime**, **prezime** (**std::string**) i **placa** (**double**). Klasa **Student** dodatno ima člansku varijablu **stipendija** (**double**). Varijabla **postotak** za objekte tipa **Osoba** iznosi *0.15*, osim što za objekte tipa **Student** iznosi *0.1*. Čistu virtualnu funkciju za **PO** odredite sami.

Implementirajte konstruktore bez parametara i konstruktore s parametrima za sve tri klase (konstruktori s parametrima primaju sve varijable osim varijable **postotak** za klase **Osoba** i **Student**). Implementirajte i destruktore za sve tri klase. Konstruktor/destruktor za svaku klasu ispisuje *„Stvoren/Unisten <klasa>.\n“*, ovisno o imenu klase. Osigurajte da se pozovu svi destruktori nadređenih klasa prilikom uništavanja izvedenih klasa.

Implementirajte člansku funkciju **info()** koja ispisuje na ekran stanje svih 2, 5, odnosno 6 članskih varijabli za objekt klase **PO**, **Osoba** odnosno **Student**. Klasi **Student** implementirajte člansku funkciju **PovecajStipendiju** koja tom objektu povećava stipendiju za *M/N * 100*, gdje je *M* broj poreznih obveznika u cijelom programu kojima **oib** završava parnom znamenkom, a *N* broj studenata u glavnom programu.

Implementirajte globalnu funkciju **NaplataPoreza**(**std::list** <**PO***> L) koja vraća ukupan porez svih objekata u listi L. Svaki objekt plaća porez po formuli *postotak*placa* (stipendija se ne oporezuje).

Napišite i glavni program koji testira sve funkcionalnosti.

Sve varijable osim jedne static varijable tipa **std::list** <**PO***> moraju biti *private*. Izbjegavajte dupliciranje dijelova koda - u protivnom možete izgubiti bodove! Zabranjeno je koristiti globalne varijable i *friend* funkcije! Potrebno je pobrinuti se da se *const* koristi svugdje gdje je moguće. Smijete implementirati i dodatne članske funkcije i varijable, no i dalje sve članske varijable trebaju biti *private*. Osigurajte da oslobodite svu dinamički alociranu memoriju.

PREZIME I IME: MATIČNI BROJ STUDENTA: *BODOVI:*

Drugi zadatak (15 bodova)

Implementirajte klasu **Proizvod** koja sadrži **std::string** *ime* s imenom proizvoda i varijablu tipa **int** *masa*, koji reprezentira masu proizvoda u kilogramima. Klasa sadrži konstruktor koji prima *ime* i *masu* proizvoda te stvara novu instancu tog tipa. Kažemo da su dva proizvoda jednaka ukoliko imaju identično ime i istu masu. Klasa **Kamion** sadrži listu pokazivača na proizvode koje kamion prevozi. Listu ćemo zvati *proizvodi*. Uz to klasa sadrži varijablu *kapacitet* tipa **int** koja definira maksimalan kapacitet kamiona. Klasa mora imati konstruktor koji prima varijablu tipa **std::list** koja sadrži pokazivače na proizvode, varijablu tipa **int** koja definira kapacitet, te konstruktor koji prima varijablu tipa **int** i definira kapacitet (u tom slučaju je lista proizvoda prazna).

Implementirajte:

- Operator +, koji od dva ulazna kamiona k1 i k2 stvara novi kamion s kapacitetom k1.kapacitet + k2.kapacitet, te u svojoj listi predmeta sadrži pokazivače na kopije svih predmeta sadržanih u kamionima k1 i k2, sortirane uzlazno po masi proizvoda.
- Operator -, koji od dva ulazna kamiona k1 i k2 stvara novi kamion koji ima kapacitet min(k1.kapacitet, k2.kapacitet). Predmeti se dodaju u novi kamion na način da se mase predmeta smještenih u kamione k1 i k2 uspoređuju po parovima te se stvori kopija predmeta s manjom masom u paru i odgovarajuća adresa spremi u listu predmeta novog kamiona.
- Operator +=, koji redom preuzima proizvode kamiona k dok se ne popuni kapacitet kamiona nad kojim je pozvan. Ukoliko kamion nad kojim je pozvan operator može preuzeti sve proizvode kamiona k, tada kamion k više ne prevozi niti jedan predmet (prazan je).
- Operator -=, koji redom prebacuje proizvode iz kamiona nad kojim je pozvan u kamion k do popunjenja kapaciteta k.kapacitet. Ukoliko kamion k može preuzeti sve predmete koje prevozi kamion nad kojim je pozvan operator, tada kamion nad kojim je pozvan operator postaje prazan (ne prevozi niti jedan predmet).
- Implementirajte operator ispisa koji omogućava ispis kamiona na konzolu. Ispis mora sadržavati kapacitet kamiona i imena te mase svih proizvoda koje sadrži.
- Operator pretvorbe kamiona u int, koji vraća ukupnu masu predmeta koje kamion prevozi.

Napišite generičku parametriziranu funkciju **prebroji** koja prima iteratore na početak i kraj intervala spremnika koji sadrži kamione, te unarni predikat *pred*. Funkcija vraća broj kamiona u spremniku koji zadovoljavaju svojstvo definirano unarnim predikatom *pred*.

Napravite 2 verzije unarnog predikata, **pred1** koji vraća *istina* ako kamion nije prazan, inače vraća *laž*. **pred2** koji vraća *istina* ako kamion sadrži barem 10 predmeta mase veće od 100 kg, inače vraća *laž*.

Napomene:

- Ulazne i izlazne parametre svih operatora i funkcija **definirajte sami**.
- Osigurajte da **ne dolazi** do kopiranja memorije prilikom prijenosa parametara funkcijama i operatorima (osim ako to nije nužno).
- **Onemogućite mijenjanje** svih objekata proslijeđenih funkcijama, itd. ukoliko njihovo mijenjanje nije nužno za implementaciju navedenih funkcionalnosti.

- **Pazite** da se pri uništenju klasa dealociraju svi objekti koji se više ne koriste.
- Operatori + i – klase **kamion** moraju raditi ispravno u izrazima oblika 1 + k, gdje je k varijabla tipa **kamion**.
- Operatori += i -= moraju omogućiti **ulančavanje!**
- **Onemogućite pristup svim podacima** (*ime, masa, proizvodi, kapacitet*) izvan klase.
- Dozvoljeno je pisanje pomoćnih funkcija.
- **Zabranjeno je** koristiti funkcije i algoritme definirane u zaglavlju <*algorithm*>

PREZIME I IME:

JMBAG:

BODOVI: