Zadaci za Laboratorijsku vježbu 8

NAPOMENA: Studenti bi trebali da razmisle o zadacima koji će se raditi na laboratorijskoj vježbi prije nego što dođu na vježbu, tako da već u startu imaju osnovne ideje kako riješiti zadatke. U suprotnom, rad na vježbi neće biti produktivan. Zadatke koje studenti ne stignu uraditi za vrijeme vježbe, trebali bi ih samostalno uraditi kod kuće.

1. Definirajte strukturu "`Vrijeme`" koja sadrži tri polja "`sati`", "`minute`" i "`sekunde`", i koja predstavlja tekuće vrijeme u toku dana, izraženu u satima, minutama i sekundama. Zatim definirajte funkciju "`IspisiVrijeme`" koja ispisuje vrijeme proslijeđeno kao parametar u funkciju u obliku *hh*:*mm*:*ss* (tj. sati, minute i sekunde se ispisuju kao dvocifreni brojevi, eventualno sa vodećim nulama), kao i funkciju "`SaberiVrijeme`" koja prima dva vremena kao parametre, i vraća kao rezultat treće vrijeme koje nastaje sabiranjem vremena koja su proslijeđena kao parametri (npr. sabiranjem vremena 3 h 34 min 52 sek i 4 h 42 min 20 sek treba da se dobije vrijeme 8 h 17 min 12 sek). Ukoliko se funkciji "`IspisiVrijeme`" pošalje struktura koja sadrži neispravno vrijeme (tj. ukoliko sati nisu u opsegu od 0 do 23 a minute i sekunde od 0 do 59 uključivo), treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Neispravno vrijeme". Isto se treba desiti i ukoliko bilo koji od dva parametra funkcije "`SaberiVrijeme`" predstavlja neispravno vrijeme. Da biste ovo lakše izveli, napišite i pomoćnu funkciju "`TestirajVrijeme`" koja prima vrijeme kao parametar. Ukoliko je vrijeme legalno ova funkcija ne radi ništa, a ukoliko nije, baca gore navedeni izuzetak (ovu funkciju ćete pozivati iz funkcija "`IspisiVrijeme`" i "`SaberiVrijeme`"). Konačno, napravite mali testni program u kojem ćete testirati napisane funkcije. Dijalog između programa i korisnika treba izgledati poput sljedećeg:

```
Unesite prvo vrijeme (h m s): 3 34 52
Unesite drugo vrijeme (h m s): 4 42 20
Prvo vrijeme: 03:34:52
Drugo vrijeme: 04:42:20
Zbir vremena: 08:17:12
```

   Ukoliko neko od unesenih vremena nije legalno, program treba da se odmah prekine po okončanju spornog unosa, uz odgovarajući prikaz poruke, kao u sljedećem dijalogu:

```
Unesite prvo vrijeme (h m s): 13 179 66
Neispravno vrijeme
```

   Radi jednostavnosti, pretpostavite da će se sa tastature unositi samo cijeli brojevi, odnosno nije potrebno provjeravati da li je korisnik zaista unio cijeli broj kada se to od njega očekuje.

2. Prepravite program za obradu učenika sa Predavanja 8_a, koji koristi vektor pokazivača na dinamički alocirane objekte tipa "`Ucenik`" tako da se umjesto običnih pokazivača koriste pametni pokazivači (obični pokazivači se ne smiju pojaviti nigdje u programu). Upute za ovu prepravku date su na predavanjima.

3. Prepravite program za obradu učenika sa Predavanja 8_a, koji koristi vektor pokazivača na dinamički alocirane objekte tipa "`Ucenik`" tako da se umjesto vektora pokazivača na učenike koristi dinamički alocirani niz (običnih) pokazivača na učenike (biblioteku i tip "`vector`" nije dozvoljeno koristiti). Upute za ovu prepravku date su na predavanjima.

4. Napišite funkciju "`ZamijeniPremaRjecniku`" koja određene riječi u nekom stringu zamjenjuje nekim drugim riječima. Prvi parametar je tipa "`string`" i on predstavlja string u kojem se vrši zamjena. Drugi parametar je mapa (tj. objekat tipa "`map`") kod koje su i ključno polje i pridružena vrijednost tipa "`string`". Ova mapa se sastoji od uređenih parova, u kojima prvi element predstavlja riječ koja se zamjenjuje, a drugi element predstavlja zamjensku riječ koja se treba ubaciti umjesto riječi koja se zamjenjuje. Funkcija treba kao da rezultat vrati novi string koji se dobija nakon izvršenih zamjena. Na primjer, sljedeći testni primjer treba da ispiše "how yes no":

```
std::map<std::string, std::string> moj_rjecnik{{"jabuka", "apple"}, {"da", "yes"},
  {"kako", "how"}, {"ne", "no"}, {"majmun", "monkey"}};
std::cout << ZamijeniPremaRjecniku("kako da ne", moj_rjecnik);
```

   Radi jednostavnosti, pretpostavite da string koji se transformira sadrži samo mala slova i razmake (i ništa drugo). Isto tako, pretpostavite da riječi u parovima koje čine mapu koja predstavlja rječnik

sadrže samo mala slova, bez razmaka. Ove pretpostavke ne morate provjeravati, već je samo uzmite kao takvu. Međutim, vodite računa da je moguće da riječi budu razdvojene sa više razmaka, kao i da se na početku odnosno kraju stringa mogu također nalaziti razmaci. Drugim riječima, string poput " kako da ne " uz pomoć istog "rječnika" kao i gore navedenom primjeru treba da se prevede u " how yes no ".

Treba paziti da se uvijek se zamjenjuju samo cijele riječi, nikad njihovi dijelovi. Na primjer, ukoliko string koji se transformira glasi "davor martić ima sestre maju i juliju" i ukoliko se mapa koja definira rječnik sastoji od parova ("mart", "ožujak"), ("maj", "svibanj") i ("jul", "srpanj"), rezultat transformacije treba da bude isti string bez ikakve promjene, odnosno rezultat ne treba da bude string "davor ožujakić ima sestre svibanju i srpanjiju". Inače, nešto slično tome desilo se prije nekoliko godina prilikom pokušaja automatskog "prevođenja" transkripta jedne parlamentarne sjednice sa bosanskog na hrvatski jezik uz pomoć Word-ove "Find & Replace" komande, kada je poslanik Martić postao Ožujakić. Interesantno je što bez obzira što Word ima opciju "Whole words only" kojom bi se izbjegao taj problem, "prevodioci" su je držali isključenom, jer su smatrali da na taj način "rješavaju" problem promjene riječi po padežima, koje inače komanda "Find & Replace" ne može da "uhvati".

Napišite i kratki testni program u kojem ćete demonstrirati napisanu funkciju. Način komunikacije između korisnika i programa izvedite prema vlastitoj volji (neće se testirati).

5. U modernom programiranju se, umjesto vlastito definiranih struktura, sve češće koristi standardni bibilotečki tip "`tuple`" (uređena *n*-torka). Zaista, s tim tipom može se raditi sve što i sa klasičnim vlastito definiranim strukturama (pa i još ponešto), uz jedinu bitnu razliku da polja unutar uređene n-torke nemaju imena, nego da im se pristupa po rednom broju. Tako, recimo, u primjeru iz Predavanja 8_a, umjesto da tip "`Vektor3d`" definiramo kao vlastito definiranu strukturu, možemo ga definirati kao uređenu trojku čija su sva tri polja tipa "`double`". Da ne bismo stalno pisali nešto poput "`std::tuple<double, double, double>`", možemo uvesti "`typedef`" deklaraciju

```
typedef std::tuple<double, double, double> Vektor3d;
```

nakon čega možemo koristiti prosto "`Vektor3d`" kao ime tipa. Prepravite sve funkcije u programu iz primjera o kojem je riječ da rade sa ovako izmijenjenom definicijom tipa "`Vektor3d`", uz ograničenje da glavni program (funkcija "`main`") treba da ostane potpuno neizmijenjen. Pri tome, barem u jednoj od funkcija koju prepravljate upotrijebite i funkciju "`tie`", tamo gdje mislite da bi njena upotreba mogla biti od najveće koristi (ovo je čisto da demonstrirate da razumijete mogućnost njene primjene).

6. Definirajte generičku strukturu "`Cvor`" koja sadrži polje "`element`" neodređenog tipa koji će se kasnije specificirati i polje "`veza`" koje je tipa pokazivač na generičku strukturu "`Cvor`" sa istim tipom polja "`element`" kao što je i ona sama. Zatim napišite pet jednostavnih generičkih funkcija sa sljedećim prototipovima:

```
template <typename TipElemenata>
  Cvor<TipElemenata> *KreirajPovezanuListu(TipElemenata zavrsni);
template <typename TipElemenata>
  int BrojElemenata(Cvor<TipElemenata> *pocetak);
template <typename TipElemenata>
  TipElemenata SumaElemenata(Cvor<TipElemenata> *pocetak);
template <typename TipElemenata>
  int BrojVecihOd(Cvor<TipElemenata> *pocetak, TipElemenata prag);
template <typename TipElemenata>
  void UnistiListu(Cvor<TipElemenata> *pocetak);
```

Funkcija "`KreirajPovezanuListu`" čita podatke tipa "`TipElemenata`" sa tastature sve dok se ne unese podatak čija je vrijednost jednaka vrijednosti parametra "`zavrsni`", uvezuje te elemente u povezanu listu čvorova tipa "`Cvor`" (sa elementima tipa "`TipElemenata`") pri čemu se završni element ne uvezuje, te vraća kao rezultat pokazivač na prvi čvor tako kreirane liste (ili nul-pokazivač ako nije unesen niti jedan koristan element). Funkcija "`BrojElemenata`" prima kao parametar pokazivač na prvi čvor povezane liste čvorova, a vraća kao rezultat broj elemenata tako povezane liste (ili 0 ako se zada nul-pokazivač). Funkcija "`SumaElemenata`" također prima kao parametar pokazivač na prvi

čvor povezane liste čvorova, a vraća kao rezultat sumu svih elemenata tako povezane liste (za elemente se pretpostavlja da je definirana operacija sabiranja). Ukoliko je parametar nul-pokazivač, vraća se podrazumijevana vrijednost za tip "`TipElemenata`". Slična je i funkcija "`BrojVecihOd`", samo što ona vraća kao rezultat broj elemenata liste koji su veći od vrijednosti parametra "`prag`" (pri tome se pretpostavlja da je za elemente definirana operacija poređenja pomoću operatora "`>`"). Ukoliko je parametar nul-pokazivač, vraća se 0 kao rezultat. Konačno, funkcija "`UnistiListu`" oslobađa memoriju koju zauzimaju svi čvorovi povezane liste čvorova, pri čemu kao i u prethodnim funkcijama parametar pokazuje na prvi čvor liste. Ukoliko je parametar nul-pokazivač, funkcija ne radi ništa. Iskoristite ove funkcije u programu koji čita slijed realnih brojeva sa tastature sve dok se ne unese 0, uvezuje te elemente u povezanu listu čvorova čiji su elementi realni brojevi (tj. tipa "`double`"), zatim prolazi kroz listu čvorova sa ciljem da prebroji i ispiše koliko ima elemenata u listi koji su veći od aritmetičke sredine svih unesenih elemenata, te na kraju oslobađa svu memoriju koju su zauzeli čvorovi. Dijalog između programa i korisnika treba izgledati poput sljedećeg:

```
Unesite slijed brojeva (0 za kraj): 2.5 3.17 5.8 2 -1 3.9 6 1.17 1 -2.25 9 0
U slijedu ima 5 brojeva vecih od njihove aritmeticke sredine
```