POPRAVNI ISPIT IZ PREDMETA "TEHNIKE PROGRAMIRANJA" (I PARCIJALNI)

Zadatak 1

Utvrdite šta će ispisati sljedeći program (napomenimo da ASCII kod znaka '2' glasi 50). Odgovor mora biti obrazložen:

```
#include <iostream>
#include <iomanip>
#include <cstring>
#include <complex>
#include <string>
using namespace std;
int f(char x) { return x; }
int f(int x) { return x + 1; }
int f(float x) { return 7 / x; }
int f(double x) { return x; }
int f(const char x[]) { return strlen(x); }
int f(string x) { return x.size() + 3; }
template <typename Tip>
int g(Tip x) { return f(x); }
int main() {
  string s("2");
  cout << g(2) << g<float>(2) << g(2.) << g('2') << g<int>('2') << endl;
  cout << g("2") << g(s) << g("222") << g<string>("222") << endl;
  int a(2), b(a), &c(a);
  const int d(a), &e(a), &f(a + 0);
  a += 2; cout << a << b << c << d << e << f << endl;
  double u(2 * 2); complex<double> v(u), w(-v);
  cout << sqrt(u) << setw(7) << sqrt(v) << " " << sqrt(w) << endl;
  return 0;
}
```

Zadatak 2

Napišite funkciju koja kao parametar prima vektor cijelih brojeva, a koja kao rezultat vraća novi vektor čiji su elementi faktorijeli odgovarajućih elemenata vektora koji je zadan kao parametar. Na primjer, ukoliko se funkciji proslijedi vektor čiji su elementi 6, 3, 2, 7, 1, 0, 4 i 5, funkcija treba da kao rezultat vrati vektor čiji su elementi 720, 6, 2, 5040, 1, 1, 24 i 120. Ukoliko je neki od elemenata vektora negativan, funkcija treba da baci izuzetak. Napišite i kratki isječak programa u kojem ćete demonstrirati kako se koristi napisana funkcija (obavezno predvidite i hvatanje eventualno bačenog izuzetka).

Zadatak 3

Napišite generičku funkciju "`UnosBroja`" sa tri parametra. Funkcija treba da omogući pouzdano unošenje brojeva u program, uz potpunu kontrolu grešaka pri unosu. Prvi i drugi parametar su tipa "`string`". Pri tome, prvi parametar predstavlja tekst koji se ispisuje korisniku kao obavijest da treba unijeti broj (prompt), dok drugi parametar predstavlja tekst koji se ispisuje korisniku kao upozorenje u slučaju da unos nije ispravan. Treći parametar je referenca na proizvoljni numerički tip, a predstavlja promjenljivu u koju će se smjestiti uneseni broj. Na primjer, funkcija se može pozvati na sljedeći način:

```
UnosBroja("Unesi prvi broj: ", "Neispravan unos!\n", prvi_broj);
```

Funkcija treba da traži unos od korisnika sve dok unos ne bude ispravan. Napišite i kratak isječak programa koji demonstrira kako se koristi napisana funkcija.

Zadatak 4

Napišite funkciju "`AnalizaStringa`" sa četiri parametra. Prvi parametar je neki dinamički string (tj. objekat tipa "`string`"). Funkcija treba da utvrdi koliko u tom stringu ima znakova koji su velika slova, zatim znakova koji su mala slova, kao i ostalih znakova (tj. znakova koji nisu slova), i da smjesti rezultate analize redom u drugi, treći i četvrti parametar respektivno. Napišite i kratki isječak programa u kojem ćete demonstrirati kako bi se mogla upotrijebiti napisana funkcija.

Zadatak 5

Napišite generičku funkciju koja u bloku elemenata između pokazivača "`p1`" i "`p2`" pronalazi element za koji funkcija "`f`" daje najmanju moguću vrijednost i kao rezultat vraća pokazivač na taj element (ukoliko takvih elemenata ima više, funkcija treba da vrati pokazivač na prvi takav element). Parametri funkcije su "`p1`" i "`p2`" i "`f`" (takva funkcija ne postoji u biblioteci "`algorithm`", ali je potpuno "u duhu" funkcija iz ove biblioteke). Funkcija bi trebala biti zasnovana na potpunoj dedukciji tipova, tako da se umjesto pokazivača mogu koristiti i iteratori (ukoliko ne znate izvesti potpunu dedukciju, koristite parcijalnu dedukciju, samo ćete time izgubiti 1 poen). Napišite i mali isječak programa u kojem ćete demonstrirati kako biste ovu funkciju primijenili da iz nekog fiksnog niza od 10 realnih brojeva ispišete onaj element za koji funkcija $f(x) = (x - \sin x) / (x + \cos x)$ ima najmanju vrijednost

Zadatak 6

Napišite funkciju koja ima dva parametra, od kojih je prvi niz realnih brojeva, a drugi broj elemenata u nizu. Funkcija treba da dinamički alocira novi niz realnih brojeva koji sadrži isti broj elemenata kao i zadani niz, da prepiše sve elemente niza u kreirani niz u obrnutom poretku (tj. prvi element izvornog niza treba da postane posljednji element novog niza, drugi element izvornog niza treba da postane pretposljednji element novog niza, itd.) i da vrati pokazivač na prvi element tako kreiranog niza kao rezultat. U slučaju da alokacija ne uspije, funkcija treba da baci izuzetak "Alokacija nije uspjela". Napišite i mali isječak programa u kojem ćete sa tastature unijeti niz od 10 elemenata, pozvati napisanu funkciju da kreira izvrnuti niz, zatim ispisati elemente tako formiranog niza i na kraju osloboditi memoriju koju je zauzeo novokreirani niz. Obavezno predvidite i hvatanje eventualno bačenog izuzetka.

Zadatak 7

Pretpostavimo da je potrebno sortirati niz brojeva uz pomoć funkcije "`sort`" u takav poredak da brojevi sa manje cifara dođu ispred brojeva sa više cifara, ali da među brojevima sa jednakim brojem cifara veći brojevi dolaze ispred manjih brojeva. Na primjer, broj 23 treba doći ispred broja 4387 (jer broj 23 ima manje cifara od broja 4387), ali među brojevima 5428 i 4387 koji imaju jednak broj cifara, broj 5428 treba doći ispred broja 4387. Napišite funkciju kriterija koja omogućava da funkcija "`sort`" obavi ovakvo sortiranje. Napisanu funkciju demonstrirajte u testnom programu u kojem se sa tastature unosi 10 cijelih brojeva u neki dek, nakon čega se vrši njihovo sortiranje u skladu sa opisanim kriterijem te ispis tako sortiranog deka na ekran.

POPRAVNI ISPIT IZ PREDMETA "TEHNIKE PROGRAMIRANJA" (II PARCIJALNI)

NAPOMENA: Sve metode implementirajte izvan deklaracije klase, osim trivijalnih metoda koje trebate implementirati direktno unutar deklaracije klase. Sve metode koje su inspektori, obavezno deklarirajte kao takve. Ukoliko su Vam potrebne pomoćne metode koje nisu predviđene interfejsom klase, možete ih kreirati, ali u privatnom dijelu klase.

Zadatak 1

Elektrotehničkom fakultetu je potreban interni telefonski imenik svojih uposlenika. Stoga je potrebno razviti kontejnersku klasu "`Imenik`" koja će čuvati kolekciju podataka o uposlenicima i njihovim (lokalnim) telefonskim brojevima. Prva ideja je bila da se podaci o jednom uposleniku čuvaju u strukturi "`Uposlenik`" koja kao svoje atribute sadrži ime uposlenika zajedno sa prezimenom (tipa niza od max. 20 znakova), adresu (tipa niza od max. 50 znakova) i lokalni broj telefona (cjelobrojnog tipa). Dalje je planirano da se podaci o svim uposlenicima čuvaju u dinamički alociranom nizu čiji su elementi gore opisane strukture (kojem se pristupa putem odgovarajućeg atrubuta klase).

Za kreiranje primjeraka klase, predviđeno je postojanje konstruktora sa jednim parametrom koji vrši neophodnu dinamičku alokaciju, pri čemu parametar predstavlja maksimalni broj uposlenika koji se mogu evidentirati. Ovaj konstruktor se ne smije koristiti za automatsku konverziju cjelobrojnih podataka u tip ove klase. Predviđen je i destruktor koji se brine o oslobađanju svih resursa koje su primjerci ove klase zauzeli tokom svog života. Radi bezbjednog kopiranja i međusobnog dodjeljivanja primjeraka ove klase, planirano je da se podrže i konstruktor kopije te preklopljeni operator dodjele, koji će se zasnivati na strategiji dubokog kopiranja.

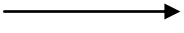
Za dodavanje novih podataka u imenik predviđene su tri metode istog imena "`DodajUposlenika`". Prva metoda kao parametre prima podatke o uposleniku (ime i prezime, adresa i broj telefona). U slučaju da su podaci o imenu i prezimenu ili adresi predugi, baca se izuzetak. Druga metoda kao jedini parametar ima konstantnu referencu na neki objekat tipa "`Uposlenik`" koja sadrži sve podatke o uposleniku koji se upisuje, dok treća metoda umjesto konstantne reference prima pokazivač na odgovarajući objekat tipa "`Uposlenik`" (u ovom trećem slučaju, klasa "`Imenik`" ne preuzima vlasništvo nad objektom, tj. nije odgovorna za njegovo eventualno brisanje). Sve tri metode bacaju izuzetak ukoliko se dostigne maksimalni broj uposlenika koji se mogu evidentirati.

Za pretragu imenika predviđene su metoda "`IspisiUposlenika`" koja ispisuje podatke o uposleniku čije se ime i prezime zadaje kao parametar, metoda "`IspisiSveNaSlovo`" koja ispisuje podatke o svim uposlenicima čije ime počinje slovom koje se zadaje kao parametar, te metoda "`IspisiImenik`" koja ispisuje čitav telefonski imenik sortiran po abecednom poretku. Predviđen je i preklopljeni operator "`[]`" koji vraća referencu na broj telefona uposlenika čije se ime zadaje unutar uglastih zagrada (referenca se vraća sa ciljem da se omogući izmjena broja). U slučaju da se ovaj operator primijeni na konstantni objekat, umjesto reference treba vratiti kopiju broja.

Klasa "`Imenik`" također treba da podržava unarni operator "`!`" koji vraća logičku vrijednost "**`true`**" ukoliko je imenik prazan, a "**`false`**" u suprotnom, te binarni operator "`+`" koji sastavlja dva imenika u jedan. Preciznije, ukoliko su "`X`" i "`Y`" dva imenika (tj. objekta tipa "`Imenik`"), "`X + Y`" treba da bude novi imenik koji u sebi sadrži sve podatke kako iz imenika "`X`", tako i iz imenika "`Y`".

Konačno, za trajno čuvanje podataka pohranjenih u imeniku predviđene su i metode "`Sacuvaj`" i "`Obnovi`" koje snimaju čitav telefonski imenik u binarnu datoteku čije se ime zadaje kao parametar, odnosno obnavljaju njegov sadržaj iz binarne datoteke čije se ime zadaje kao parametar.

Vaš zadatak je da dizajnirate klasu "`Imenik`" u skladu sa zadatim specifikacijama.



Zadatak 2

Mada klasa koju ste razvili u prethodnom zadatku radi jako lijepo i ispunjava postavljene zahtjeve, ispostavilo se da ona neracionalno koristi računarske resurse, te da bi bolje bilo u dinamički alociranom nizu čuvati ne same objekte tipa "`Uposlenik`" nego pokazivače na takve objekte, pri čemu bi se individualni objekti tipa "`Uposlenik`" alocirali dinamički. Napišite ponovo klasu "`Imenik`" u skladu sa ovako izmijenjenom organizacijom, ali tako da interfejs ostane isti, odnosno da korisnik klase ne primijeti da je došlo do promjene u internoj organizaciji klase.

NAPOMENA: Kod metode "`DodajUposlenika`" koja kao parametar prima pokazivač na objekat tipa "`Uposlenik`", sami odlučite da li će klasa "`Imenik`" preuzeti vlasništvo nad objektom na koji pokazivač pokazuje ili ne. Mada je kod kontejnerskih klasa ovog tipa (tj. onih koje u sebi interno čuvaju pokazivače na objekte) uobičajeno da klasa *preuzima* vlasništvo nad objektom na koji pokazivač pokazuje, takva odluka dovodi do izvjesne semantičke razlike (tj. razlike u ponašanju) ove metode u odnosu na to kako je ona radila u Zadatku 1 (s obzirom da tamo preuzimanje vlasništva nije bilo prirodno), tako bi bilo bolje te semantičke razlike izbjeći. Nažalost, to nije posve jednostavno ispravno izvesti, tako da Vam je dopušteno da ovdje odstupite od toga (u praksi, takve semantičke razlike bi morale biti propisno dokumentirane da korisnik klase ne bude iznenađen).

Zadatak 3

Obje razvijene klase u zadacima 1 i 2 rade dobro, ali su pomalo "staromodne", jer ne koriste modernije koncepte jezika C++. Zbog toga je odlučeno da se klasa "`Imenik`" iz zadatka 1 modernizira. Prva izmjena je što će, radi bolje enkapsulacije, "`Uposlenik`" sada biti klasa a ne struktura, pri čemu će njeni atributi koji čuvaju ime i prezime odnosno adresu sada biti dinamički stringovi (tj. tipa "`string`") a ne obični nizovi znakova. Pored ovih atributa, klasa "`Uposlenik`" će sadržavati konstruktor koji inicijalizira sve atribute u skladu sa parametrima konstruktora, te trivijalne pristupne metode "`DajIme`", "`DajAdresu`" i "`DajTelefon`" pomoću kojih se mogu saznati vrijednosti odgovarajućih atributa.

Druga izmjena je što će se podaci o uposlenicima umjesto u dinamički alociranom nizu objekata tipa "`Uposlenik`" čuvati u vektoru objekata tipa "`Uposlenik`". Ovim otpada potreba da se putem konstruktora zadaje maksimalan broj uposlenika koji se mogu registrirati. Također je odlučeno da metode "`Sacuvaj`" i "`Obnovi`" koriste tekstualne datoteke a ne binarne, pri čemu možete sami odlučiti kako će izgledati format zapisa u datoteci (prelaskom na tekstualne umjesto binarnih datoteka ujedno rješavamo i problem uzrokovan činjenicom da vektori i stringovi nisu POD tipovi podataka, a opće je poznato da ne-POD tipovi podataka i binarne datoteke baš i nisu "u ljubavi"). Napišite ponovo klasu "`Imenik`" u skladu sa ovako izmijenjenom organizacijom, ali tako da interfejs ostane isti, odnosno da korisnik klase ne primijeti da je došlo do promjene u internoj organizaciji klase u odnosu na klasu iz zadatka 1 (osim izmjene koja je uzrokovana gubitkom parametra konstruktora).

Zadatak 4

Mada i prethodna klasa također radi izvrsno, u šta ne treba ni sumnjati, ona posjeduje isti problem po pitanju korištenja računarskih resursa kao i klasa iz zadatka 1. Naime, bolje bi bilo da se u vektoru ne čuvaju sami objekti tipa "`Uposlenik`" nego pokazivači na takve objekte, pri čemu se individualni objekti tipa "`Uposlenik`" alociraju dinamički. Napišite ponovo klasu "`Imenik`" u skladu sa ovako izmijenjenom organizacijom, ali tako da interfejs ostane isti kao u prethodnom zadatku, odnosno da korisnik klase ne primijeti da je došlo do promjene u internoj organizaciji u odnosu na klasu iz prethodnog zadatka (a ni u odnosu na klasu iz zadatka 2 osim izmjene koja je uzrokovana gubitkom parametra konstruktora).

NAPOMENA: Za ovu klasu u odnosu na zadatak 3 vrijedi ista napomena kao u zadatku 2 u odnosu na klasu iz zadatka 1.

Zadaci namjerno nisu striktno bodovani. O tome koliko ćete poena dobiti i da li ćete položiti ili ne prvenstveno ovisi o Vašoj sposobnosti da razumijete šta se i kako mijenja u pojedinim realizacijama načelno istog problema (položiće svi oni koji ispravno uoče i ispravno realiziraju DOVOLJNO izmjena). Ovi faktori su važniji od same količine napisanog.