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

Zadatak 1. (5 poena)

Prikažite *tačan izgled ekrana* na kraju izvršavanja ovog C++ programa, uz kratko obrazloženje zbog čega su rezultati onakvi kakvi jesu. Bitan je *svaki razmak* i *svaki prelazak u novi red*.

```
#include <iostream>
using namespace std;
void P(int &a, int b, int &c) {
  a = b * c; b = a * c; c = a * b;
  cout << a << " " << b << " " << c << endl;
}
void Q(int a, int &b, int c) {
  a = b * c; b = a * c; c = a * b;
  cout << a << " " << b << " " << c << endl;
}
int main() {
  int b(3); P(b, b, b); cout << b << endl;
  b = 2; Q(b, b, b); cout << b << endl;
  int a1(1); int a2(a1); const int a3(a1);
  int &a4(a1); const int &a5(a1); const int &a6(a1 + 0);
  a1 += 5; cout << a1 << a2 << a3 << a4 << a5 << a6 << endl;
  return 0;
}
```

Zadatak 2. (2,5 poena)

Napišite funkciju sa tri parametra koja računa zbir svih cifara na parnim pozicijama svih cifara na neparnim pozicijama cijelog broja koji joj je naveden kao prvi parametar, i smješta rezultate u drugi i treći parametar respektivno (npr. u broju 3742961 cifre 7, 2 i 6 su na parnim, a cifre 3, 4, 9 i 1 na neparnim pozicijama). Napišite i mali isječak programa u kojem ćete demonstrirati kako se upotrebljava napisana funkcija.

Zadatak 3. (3 poena)

Za neki slijed brojeva kažemo da je *oscilatoran* ukoliko mu elementi naizmjenično rastu i opadaju ili obrnuto. Na primjer, slijed brojeva 3, 7, 4, 6, 1, 5, 2, 3 je oscilatoran, kao i slijed 3, 1, 5, 4, 7, 2, 3, 2. Napišite funkciju koja prima kao parametar vektor realnih brojeva a koja vraća kao rezultat logičku vrijednost "tačno" ili "netačno" u ovisnosti da li je slijed brojeva pohranjen u vektoru oscilatoran ili ne.

Zadatak 4. (3 poena)

Napišite generičku funkciju "`BrojZajednickih`" sa četiri parametra "`p1`", "`p2`", "`p3`" i "`p4`" koji su "u duhu" parametara funkcija iz biblioteke "`algorithm`". Parametri "`p1`" i "`p2`" omeđuju jedan blok podataka (tj. "`p1`" pokazuje na početak bloka a "`p2`" tačno iza kraja bloka), dok "`p3`" i "`p4`" omeđuju drugi blok podataka. Elementi oba bloka su proizvoljnog ali istog tipa. Funkcija treba da kao rezultat vrati broj elemenata koji se javljaju kao zajednički elementi i u jednom i u drugom bloku. Na primjer, neka su date sljedeće deklaracije:

```
int a[8] = {3, 7, 2, 3, 1, 5, 5, 2};
int b[10] = {4, 6, 7, 8, 1, 3, 1, 6, 4, 7};
```

Tada poziv funkcije "`BrojZajednickih(a, a + 8, b, b + 10)`" treba da kao rezultat dâ broj 3, jer se tri elementa (3, 7 i 1) pojavljuju u oba niza. Funkciju bi u cijelosti trebalo napisati korištenjem pokazivačke aritmetike, tj. bez upotrebe indeksiranja. Napišite i kratki testni program u kojem ćete demonstrirati napisanu funkciju.

Zadatak 5. (2,5 poena)

Postoje razni metodi pomoću kojih je moguće izračunati približnu vrijednost integrala neke funkcije $f(x)$ na intervalu (a, b). Jedan od najjednostavnijih ali ujedno i najmanje tačnih metoda je tzv. *trapezno ili Eulerovo pravilo*, prema kojem je

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{b-a}{n}\left[\frac{1}{2}f(a)+\frac{1}{2}f(b)+\sum_{k=1}^{n-1} f\left(a+\frac{b-a}{n}k\right)\right]$$

gdje je n broj podintervala na koji dijelimo interval (a, b). Napišite funkciju "`TrapeznoPravilo`" koja prima kao parametre *f*, *a*, *b* i *n* (*f* je *funkcija čiji se integral računa*) a koja kao rezultat daje približnu vrijednost integrala računatu pomoću trapeznog pravila. Također napišite i isječak programa u kojem ćete demonstrirati kako bi se ova funkcija mogla iskoristiti za računanje integrala funkcije $\sin x$ na intervalu $(0, \pi)$, te funkcije $1/x$ na intervalu $(1, 2)$.

Zadatak 6. (4 poena)

Napišite funkciju koja kao parametar prima vektor stringova (tj. vektor čiji su elementi tipa "`string`"). Funkcija treba da podatke pohranjene u tom vektoru stringova organizira u memoriji koristeći dinamičku alokaciju memorije. Na prvom mjestu, funkcija treba da dinamički kreira niz pokazivača na znakove koji ima onoliko elemenata koliko ima stringova u vektoru. Zatim, za svaki string u vektoru treba dinamički alocirati odgovarajući prostor, prepisati sadržaj stringa u kreirani prostor, i dodijeliti adresu kreiranog prostora odgovarajućem pokazivaču u nizu pokazivača. Funkcija kao rezultat vraća pokazivač na prvi element kreiranog niza pokazivača (preko koga se može izvršiti pristup pohranjenim podacima). Ukoliko dođe do bilo kakvih problema pri alokaciji, funkcija treba da baci izuzetak, uz prethodno čišćenje svog "smeća" koje je bilo kreirano u postupku alokacije (tako da ni pod kojim uvjetima ne dođe do curenja memorije). Napišite i kratki isječak programa u kojem ćete demonstrirati kako biste sa tastature unijeli *n* stringova u vektor stringova (*n* se također unosi sa tastature), pozvali napisanu funkciju, ispisali na ekran dinamički alocirane nizove znakova koje je kreirala funkcija i, na kraju, obrisali sav alocirani prostor. Obavezno treba predviditi i hvatanje eventualno bačenih izuzetaka.

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

Zadatak 1. (8 poena)

Linearne funkcije jedne realne promjenljive predstavljaju vjerovatno najviše korištene funkcije u svim oblastima nauke i tehnike. To su funkcije koje se mogu predstaviti analitičkim izrazom oblika $f(x) = kx + l$ gdje je x realna promjenljiva, a k i l su realni parametri nazvane redom *koeficijent pravca* i *slobodni član*. Stoga je od koristi razviti opću klasu (možemo je nazvati recimo "`LinFun`") pomoću koje se mogu kreirati objekti koji se ponašaju poput linearnih funkcija sa parametrima koji se mogu zadavati prilikom kreiranja objekta (i eventualno naknadno mijenjati). Na primjer, takvi objekti mogu se koristiti kao u sljedećem primjeru:

```
LinFun f(3, 5);   // Kreira lin. funkciju nazvanu "f", f(x) = 3 x + 5
...
cout << f(2);     // Ispisuje vrijednost funkcije "f" za argument 2 (11)
```

Pored toga, predviđeno je da se sa objektima tipa "`LinFun`" mogu vršiti i različite manipulacije, u skladu sa opisom koji slijedi. Klasa "`LinFun`" kao jedine svoje atribute sadrži koeficijent pravca i slobodni član odgovarajuće linearne funkcije. Definirajte i implementirajte klasu sa navedenim osobinama, znajući da njen interfejs sadrži sljedeće elemente (sve metode koje su inspektori trebaju obavezno biti označene kao takve):

- Konstruktore, koji omogućavaju kreiranje objekata tipa "`LinFun`". Predviđeno je da se objekti ovog tipa mogu kreirati na više različitih načina. Moguće je pri kreiranju zadati dva parametra, koji redom predstavljaju koeficijent pravca i slobodni član linearne funkcije koja se kreira. Dalje, moguće je zadati samo jedan parametar, koji tada predstavlja slobodni član, dok se koeficijent pravca postavlja na nulu (time se faktički kreira konstantna linearna funkcija). Konačno, objekte tipa "`LinFun`" moguće je kreirati i bez zadavanja ikakvih parametara. U tom slučaju, i koeficijent pravca i slobodni član se postavljaju na nulu. Potrebno je podržati automatsku pretvorbu po kojoj se realni brojevi automatski mogu tretirati i kao konstantne linearne funkcije (tj. one sa koeficijentom pravca jednakim nuli). Napomena: mada se objekti tipa "`LinFun`" mogu kreirati na tri različita načina, to ne znači nužno da moraju postojati tri konstruktora – ako možete proći sa manje, tim bolje po Vas.
- Metodu sa dva parametra koja omogućava naknadnu promjenu koeficijenta pravca i slobodnog člana već definirane linearne funkcije (tj. objekta tipa "`LinFun`").
- Pristupne metode bez parametara koje omogućavaju da se saznaju vrijednosti koeficijenta pravca odnosno slobodnog člana.
- Preklopljene binarne operatore "`+`" i "`-`" koji omogućavaju da se saberu odnosno oduzmu dvije linearne funkcije, pri čemu se kao rezultati dobijaju nove linearne funkcije (njihovi koeficijenti pravca odnosno slobodni članovi jednaki su zbiru odnosno razlici koeficijenata pravca i slobodnih članova oba sabirka). Napomena: radi automatske konverzije brojeva u linearne funkcije, ovo će automatski omogućiti i sabiranje odnosno oduzimanje funkcije i broja, odnosno broja i funkcije.
- Preklopljeni unarni operator "`-`" koji daje kao rezultat negiranu linearnu funkciju, tj. funkciju čiji su koeficijent pravca i slobodni član negirani u odnosu na izvornu funkciju.
- Preklopljeni binarni operator "`*`" koji omogućava množenje linearne funkcije realnim brojem, odnosno realnog broja linearnom funkcijom. Kao rezultat se dobija nova linearna funkcija čiji su koeficijent pravca i slobodni član pomnoženi zadanim brojem. Množenje dvije linearne funkcije ne treba podržati, jer rezultat takvog množenja nije linearna funkcija.
- Preklopljeni binarni operator "`/`" koji omogućava dijeljenje linearne funkcije realnim brojem. Kao rezultat se dobija nova linearna funkcija čiji su koeficijent pravca i slobodni član podijeljeni zadanim brojem. Dijeljenje dvije linearne funkcije kao i dijeljenje realnog broja linearnom funkcijom ne treba podržati, jer rezultat takvog dijeljenja nije linearna funkcija.
- Preklopljene operatore "`+=`", "`-=`", "`*=`" i "`/=`" koji omogućavaju da izrazi poput "`X += Y`", "`X -= Y`", "`X *= Y`" i "`X /= Y`" imaju isto značenje kao i izrazi "`X = X + Y`", "`X = X - Y`", "`X = X * Y`" i "`X = X / Y`" kad god ovi imaju smisla.
- Preklopljeni operator "`++`" koji povećava slobodni član u linearnoj funkciji na koju je primijenjen za jedinicu. Potrebno je podržati i prefiksnu i postfiksnu verziju ovog operatora.

- Preklopljeni operator "`()`" koji omogućava izračunavanje vrijednosti linearne funkcije za vrijednost argumenta koja se zadaje kao parametar.
- Metodu koja daje kao rezultat novu linearnu funkciju koja je inverzna funkcija funkcije nad kojom je metoda primijenjena. Napomena: ako neka funkcija ima koeficijent pravca k i slobodni član l, njena inverzna funkcija ima koeficijent pravca $1/k$ i slobodni član $-l/k$. U slučaju dijeljenja nulom, treba baciti izuzetak.

Zadatak 2. (7 poena)

Definirajte i implementirajte klasu "`Poligon`" koja predstavlja poligon odnosno mnogougao (mnogokut) u ravni. Poligon je opisan koordinatama svojih tjemena, koje se čuvaju u dva dinamički alocirana niza realnih brojeva (jedan čuva x a drugi y koordinate tjemena), kojima se pristupa preko odgovarajućih pokazivača koje predstavljaju atribute klase. Implementacije svih metoda treba izvesti izvan klase. Interfejs klase sadrži sljedeće elemente:

- Konstruktor sa jednim parametrom, koji predstavlja broj tjemena poligona. Ovaj konstruktor, između ostalog, vrši dinamičku alokaciju prostora za pamćenje koordinata tjemena. Pored toga, koordinate svih tjemena novokreiranog poligona treba postaviti na nulu. Ne smije se dozvoliti da se preko ovog konstruktora može ostvariti automatska konverzija cijelih brojeva u objekte tipa "`Poligon`". Ukoliko je broj tjemena manji od 3, treba baciti izuzetak (jer poligoni sa manje od 3 tjemena ne postoje).
- Destruktor, koji oslobađa svu memoriju koju je objekat zauzeo tokom svog života.
- Konstruktor kopije i preklopljeni operator dodjele koji omogućavaju bezbjedno kopiranje i međusobno dodjeljivanje objekata tipa "`Poligon`" zasnovano na strategiji dubokog kopiranja.
- Metodu koja daje broj tjemena poligona.
- Metodu koja omogućava zadavanje koordinata tjemena poligona. Parametri su redni broj tjemena (u opsegu od 1 do n gdje je n ukupan broj tjemena), te x i y koordinata tjemena. Ukoliko redni broj tjemena nije ispravan, treba baciti izuzetak (vodite računa da indeksi nizova idu od nule).
- Preklopljeni operator "`[]`" koji omogućava saznavanje informacija o i-tom tjemenu poligona, pri čemu se i zadaje kao indeks. Ovaj operator kao rezultat daje objekat tipa "`Tjeme`", koji je definiran kao jednostavna struktura sa dva atributa "`x`" i "`y`" koji predstavljaju x i y koordinatu tjemena respektivno (pretpostavite da je takva struktura definirana negdje u programu).
- Metodu koja daje kao rezultat površinu poligona. Površina poligona sa n tjemena čije su koordinate (x_i, y_i), $i = 1 .. n$ računa se po formuli

  $$P = \sum_{i=3}^{n} x_1 (y_{i-1} - y_i) + x_{i-1} (y_i - y_1) + x_i (y_1 - y_{i-1})$$

- Metodu koja daje kao rezultat informaciju da li je poligon degenerisan ili nije. Poligon je degenerisan ukoliko mu makar jedan par susjednih tjemena ima jednake koordinate, inače nije.

Zadatak 3 (5 poena):

Neka je *stvar* objekat koji, između ostalog, posjeduje neku gustinu. *Lopta* je vrsta stvari koja posjeduje i poluprečnik, dok je *Cigla* vrsta stvari (oblika kvadra) koja posjeduje i dužine stranica. Razvijte hijerarhiju klasa koje opisuju ove objekte. Apstraktna bazna klasa "`Stvar`" posjedovaće atribut koji predstavlja gustinu stvaro, konstruktor koji inicijalizira ovaj atribut, čisto virtuelnu metodu koja daje zapreminu stvari, te metodu koja daje njegovu težinu (gustina pomnožena sa zapreminom). Klasa "`Lopta`" nasljeđuje se iz klase "`Stvar`", a posjeduje dodatni atribut koji predstavlja poluprečnik lopte. Njen konstruktor ima dodatni parametar (poluprečnik r) i izmijenjenu metodu za računanje zapremine, u skladu sa načinom računanja zapremine za loptu ($V = 4\pi r^3/3$). Klasa "`Cigla`" se također nasljeđuje iz klase "`Stvar`", a posjeduje dodatne atribute koji predstavljaju dužine stranica cigle. Ona također ima dodatne parametre u konstruktoru (stranice a, b i c) i implementiranu metodu za računanje zapremine ($V = abc$). Napisane klase demonstrirajte u testnom programu koji čita podatke o stvarima iz tekstualne datoteke u vektor, sortira stvari po ukupnoj težini u opadajući poredak i na kraju ispisuje težine stvari nakon sortiranja. Svaki red datoteke sadrži podatke o jednoj stvari, gdje početno slovo "`L`" označava loptu, a "`C`" ciglu. Recimo, "`L0.75 3.5`" predstavlja loptu gustine 0.75 i poluprečnika 3.5, dok "`C1.3 3.4 7 2.15`" predstavlja ciglu gustine 1.3 sa dužinom stranica 3.4, 7 i 2.15 respektivno. Datoteka sadrži isključivo ispravne podatke.