Zadaci za Laboratorijsku vježbu 1

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. Neka je dat bazen oblika kvadra dimenzije $a \times b \times c$ i keramičke pločice oblika kvadrata dimenzija $d \times d$. Napišite program koji prvo zahtijeva od korisnika da unese dužinu, širinu i dubinu bazena (tj. a, b i c) u *metrima*, kao i širinu keramičke pločice d u *centimetrima*. Program zatim treba da ispita da li je bazen moguće popločati takvim pločicama, a da se pri tome niti jedna pločica ne treba lomiti (debljinu pločice zanemariti). Ukoliko to nije moguće, treba ispisati odgovarajući komentar. Ukoliko jeste, treba ispisati koliko je pločica potrebno za popločavanje (naravno, popločavaju se zidovi i dno bazena). Slijede primjeri dva dijaloga između programa i korisnika. Radi autotestiranja, dijalozi bi trebali izgledati tačno kako je prikazano, uključujući i mjesta gdje se prelazi u novi red i prazne redove, samo se brojevi koje unosi korisnik mogu razlikovati (iza posljednje rečenice također je potreban prelazak u novi red). Afrikati poput "č", "ć" itd. se ne koriste u ispisu zbog činjenice da nije sasvim jednostavno podesiti da se oni ispravno prikazuju (to će vrijediti i ubuduće).

```
Unesite duzinu, sirinu i dubinu bazena u metrima: 5 15 3
Unesite sirinu plocice u centimetrima: 10

Za poplocavanje bazena dimenzija 5x15x3m sa plocicama dimenzija 10x10cm
potrebno je 19500 plocica.
```

```
Unesite duzinu, sirinu i dubinu bazena u metrima: 5 15 3
Unesite sirinu plocice u centimetrima: 13

Poplocavanje bazena dimenzija 5x15x3m sa plocicama dimenzija 13x13cm
nije izvodljivo bez lomljenja plocica!
```

   Za unos podataka i ispis rezultata koristite objekte "`cin`" i "`cout`" iz biblioteke "`iostream`". Koristite isključivo cjelobrojni tip podataka (tačnije tip "`int`"). Pretpostavite da su ulazni podaci smisleni (tj. ne morate testirati da li je korisnik unio smislene podatke).

2. Napišite program koji traži da se sa tastature unesu tri realna broja a, b i c, i koji ispisuje da li ta tri broja mogu biti stranice nekog trougla. Podsjetimo se da za stranice trougla mora vrijediti uvjet da su sve pozitivne i da je zbir dužina ma koje dvije stranice veći od dužine treće stranice. Ukoliko uneseni brojevi mogu predstavljati dužine stranica trougla, treba izračunati njegov obim, površinu i najmanji ugao, a zatim ispisati izračunate vrijednosti obima, površine i najmanjeg ugla, pri čemu ugao trea ispisati u stepenima, minutama i sekundama. Za računanje površine trougla koristite poznatu Heronovu formulu prema kojoj je $P = \sqrt{s(s-a)(s-b)(s-c)}$ gdje je $s = (a+b+c)/2$, a za računanje ugla koristite kosinusnu teoremu prema kojoj je $c^2 = a^2 + b^2 - 2ab\cos\gamma$ (i analogno tome za preostale kombinacije stranica i uglova). Računajte da 1 radijan ima $180/\pi$ stepeni, dok vrijednost π možete računati po formuli $\pi = 4 \cdot \operatorname{arctg} 1$ (predmetni nastavnik ne želi da vidi da neko misli da je $\pi = 3.14$, to mu pokvari dan). Ukoliko uneseni brojevi ne mogu predstavljati dužine stranica trougla, treba ispisati odgovarajući komentar. Dijalozi koje formira program trebaju izgledati poput sljedećih (zanemarite eventualne probleme sa padežima i ostalom gramatikom):

```
Unesite tri broja: 7 5 8
Obim trougla sa duzinama stranica 7, 5 i 8 iznosi 20.
Njegova povrsina iznosi 17.3205.
Njegov najmanji ugao ima 38 stepeni, 12 minuta i 47 sekundi.
```

```
Unesite tri broja: 5 15 7
Ne postoji trougao cije su duzine stranica 5, 15 i 7!
```

   Za unos podataka i ispis rezultata koristite objekte "`cin`" i "`cout`" iz biblioteke "`iostream`", a za odgovarajuća računanja funkcije iz biblioteke "`cmath`". Pretpostavite da će korisnik zaista unositi brojeve (ne nužno cijele), a ne neko "smeće".

3. Napišite program koji traži da se sa tastature unesu dva prirodna broja a i b, ne veća od 9999 i pri čemu je $a < b$, a koji zatim za sve prirodne brojeve u opsegu od a do b uključivo ispisuje tablicu njihovih kvadrata, kvadratnih korijena i prirodnih logaritama. Tačan izgled tablice vidljiv je iz dijaloga koji će biti prikazan. Uglavnom, kolona za prikaz brojeva široka je 9 polja. Brojevi se ispisuju poravnati *ulijevo*, pri čemu je prvo polje uvijek razmak. Kolone za prikaz kvadrata i korijena široke su 10 polja, dok je kolona za prikaz logaritama široka 11 polja. Kvadrati, korijeni i logaritmi se ispisuju poravnati *udesno*, pri čemu je posljednje polje uvijek razmak. Pored toga, korijeni se ispisuju fiksno na tri decimale, a logaritmi fiksno na pet decimala. Slijedi primjer kako treba izgledati dijalog između korisnika i programa:

```
Unesite pocetnu i krajnu vrijednost: 95 103

+---------+----------+----------+-----------+
| Brojevi | Kvadrati | Korijeni | Logaritmi |
+---------+----------+----------+-----------+
| 95      |     9025 |    9.747 |   4.55388 |
| 96      |     9216 |    9.798 |   4.56435 |
| 97      |     9409 |    9.849 |   4.57471 |
| 98      |     9604 |    9.899 |   4.58497 |
| 99      |     9801 |    9.950 |   4.59512 |
| 100     |    10000 |   10.000 |   4.60517 |
| 101     |    10201 |   10.050 |   4.61512 |
| 102     |    10404 |   10.100 |   4.62497 |
| 103     |    10609 |   10.149 |   4.63473 |
+---------+----------+----------+-----------+
```

   Za ispis koristite objekat "`cout`" iz biblioteke "`iostream`" i odgovarajuće manipulatore iz iste biblioteke, kao i biblioteke "`iomanip`". U programu ne trebate testirati da li su a i b zaista prirodni brojevi ne veći od 9999, i da li je $a < b$. Međutim, ispitajte kako će se program ponašati ukoliko se unesu brojevi koji ne zadovoljavaju ova ograničenja.

4. U numerologiji starih Grka, svi prirodni brojevi su se dijelili u tri kategorije, prema tome kakva im je suma svih njihovih djelilaca (ne računajući njega samog). Oni brojevi kod kojih je ta suma manja od samog broja nazivali su se *manjkavi* (engl. *deficient*), oni kod kojih je ta suma veća od samog broja nazivali su se *obilni* (engl. *abudant*), dok su oni brojevi koji su jednaki sumi svih svojih djelilaca (bez njega samog) nazivali *savršeni* (engl. *perfect*). Recimo, broj 49 je manjkav: njegovi djelioci su 1 i 7, pri čemu je $1+7=8<49$. Isto tako, broj 42 je obilan: njegovi djelioci su 1,2, 3, 6, 7, 14 i 21, pri čemu je $1+2+3+6+7+14+21=54>42$. Konačno, 28 je primjer savršenog broja: njegovi djelioci su 1, 2, 4, 7 i 14, a vrijedi $1+2+4+7+14=28$.

   Napišite program koji traži da se sa tastature unese prirodan broj n. U slučaju da korisnik unese nešto što nije prirodan broj (što uključuje i situaciju kada uneseni podatak uopće nije broj), treba ispisati poruku upozorenja, i ponoviti unos. Ukoliko je unos ispravan, program treba da ispita i ispiše da li je uneseni broj manjkav, obilan ili savršen. Nakon toga, program treba da traži unos novog broja i da ponavlja postupak sve dok se kao broj ne unese nula. Dijalog između programa i korisnika trebao bi izgledati poput sljedećeg:

```
Unesite prirodan broj ili 0 za kraj: 42
Broj 42 je obilan!
Unesite prirodan broj ili 0 za kraj: 28
Broj 28 je savrsen!
Unesite prirodan broj ili 0 za kraj: 49
Broj 49 je manjkav!
Unesite prirodan broj ili 0 za kraj: -1
Niste unijeli prirodan broj!
Unesite prirodan broj ili 0 za kraj: 2.15
Niste unijeli prirodan broj!
Unesite prirodan broj ili 0 za kraj: qwerty
Niste unijeli prirodan broj!
Unesite prirodan broj ili 0 za kraj: 0
Dovidjenja!
```