Informatikából kitűzött feladatok

I. 445. A matematikában a π nemcsak fontos természeti állandó, hanem a tudásterület egyik jelentős szimbóluma is. Minél pontosabb meghatározása ezért régóta foglalkoztatja nemcsak a matematikusokat, hanem a matematika iránt érdeklődő laikusokat is. Az évszázadok során sokféle közelítő eljárást dolgoztak ki a π meghatározására. Ezek közül Machin sorozata egy gyorsan konvergáló, tehát viszonylag kevés számú tag kiszámításával is kellően sok jegyre pontos eredményt adó megoldás. Machin a

$$\frac{\pi}{4} = 4\,\mathrm{arctg}\,\frac{1}{5} - \mathrm{arctg}\,\frac{1}{239}$$

összefüggés sorbafejtésével a

$$\frac{\pi}{4} = 4\left[\frac{1}{5} - \frac{1}{3}\left(\frac{1}{5}\right)^3 + \frac{1}{5}\left(\frac{1}{5}\right)^5 - \frac{1}{7}\left(\frac{1}{5}\right)^7 + \cdots\right] -$$

$$- \left[\frac{1}{239} - \frac{1}{3}\left(\frac{1}{239}\right)^3 + \frac{1}{5}\left(\frac{1}{239}\right)^5 - \ldots\right]$$

képletet kapta.

Készítsünk programot, amely a fenti sorozat alkalmazásával meghatározza π értékét legalább 1000 tizedesjegy pontossággal. A nagy pontosságú aritmetikai műveleteket (összeadás, kivonás, szorzás, osztás) nekünk kell megvalósítanunk. A megoldás elkészítése során legfeljebb 32 bit pontosságú egész számokat használjunk, és a ne alkalmazzuk a programozási nyelv vagy fejlesztői környezet által biztosított, 32 bitnél több jeggyel dolgozó aritmetikai műveleteket támogató modulokat sem.

A kapott eredmény ellenőrzésére a

`http://www.geom.uiuc.edu/~huberty/math5337/groupe/digits.html`

címen található adatokat javasoljuk.

Beküldendő egy `i445.zip` tömörített állományban a megoldás forráskódja, a megoldás által készített, a π első 1000 tizedesjegyét tartalmazó szöveges állomány, továbbá a megoldás során használt eljárások rövid dokumentációja.

I. 446 (É). A Rubicon egy történelmi ismeretterjesztő lap, melynek első lapszáma 1990. február 1-jén jelent meg. 2017. szeptemberéig összesen 311 számot adtak ki. Az eddig az időpontig megjelent lapszámok legfontosabb adataiból készítsünk adatbázist. Forrásként a kiadvany.txt és ajanlo.txt állományok állnak rendelkezésünkre, melyek UTF-8 kódolású szövegfájlok, az első sorok a mezőneveket tartalmazzák. Néhány számot dupla számként adtak ki, így többször szerepel két külön megjelenési hónapnál, azonos szám és különböző „azonosító” számok alatt.

1. Készítsünk új adatbázist rubicon néven. A mellékelt adatállományokat importáljuk az adatbázisba a fájlnévvel azonos nevű táblákba. Beolvasáskor állítsuk be a megfelelő típusokat és kulcsokat.

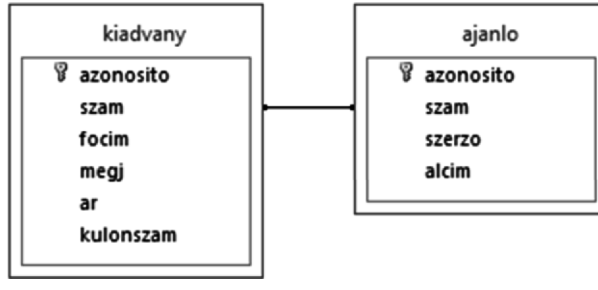
Táblák:

kiadvany (azonosito, szam, focim, megj, ar, kulonszam)

azonosito	az adott példány megjelenés azonosítója (szám), ez a kulcs;
szam	az adott kiadvány azonosítója (szám);
focim	az adott szám címe (szöveg);
megj	kiadás éve, hónapja és napja (dátum);
ar	az adott szám ára (szám);
kulonszam	az adott számot különszámként adták-e ki (logikai).

ajanlo (azonosito, szam_azon, alcimek)

azonosito	az adott cikk azonosítója (szám), ez a kulcs;
szam	az adott szám azonosítója (szám);
szerzo	az alcímhez tartozó szerző (szöveg);
alcim	az adott számhoz tartozó alcím (szöveg).



Készítsük el a következő feladatok megoldásait. Az egyes lekérdezéseknél ügyeljünk arra, hogy mindig csak a kért értékek jelenjenek meg és más adatok viszont ne. A megoldásainkat a zárójelben lévő néven mentsük el.

2. Lekérdezés segítségével adjuk meg, hogy egy számnak hány alcíme van feltüntetve az adatbázisban. Minden szám egyszer jelenjen meg, és mellette legyen látható az alcímek száma. (2alcimek)
3. Lekérdezés segítségével adjuk meg azon számok főcímét, amelyek nevében van arab szám. (3szamok)
4. Lekérdezés segítségével írassuk ki azon számok főcímét, amik különszámok voltak. (4kulonszam)
5. Mennyibe kerülne megrendelni az összes 2015-ben kiadott lapot, 2 példányban? (5ossz2015)
6. Határozzuk meg, hogy van-e, és ha igen, melyik vagy melyek azok az évek, amikor az összes hónap lapszámát lehet utánrendelni. A különszámokat ne vizsgáljuk. (6mind)
7. Számoljuk össze, hány lapszám főcímében szerepel a *szent* szó. (7szent)

8. Van-e olyan szám, amihez nem tartozik alcím az adatbázisban? Ha igen, írassuk ki a számok főcímét és kiadásának dátumát. (8nincsalcim)
9. Az alcímek között néhány helyen meg van adva a cikk szerzője. Írassuk ki azoknak a nevét, akik többször is írtak cikket. (9tobbcikk)
10. Készítsünk jelentést, amely év szerint csoportosítva, a lapszám főcíme szerinti ábécérendben jeleníti meg az adott szám árát. Dupla szám esetében két hónapnál is megjelenhet az adott szám. Az ár mögött jelenjen meg a „Ft" mértékegység. (10jelent)

Beküldendő egy tömörített `i446.zip` állományban az adatbázis, valamint egy rövid dokumentáció, amely megadja az alkalmazott adatbázis-kezelő nevét és verziószámát.

A feladat forrása:

`http://www.rubicon.hu/megrendelheto/termekek/folyoirat/`

(utolsó letöltés: 2017. 10. 01.).

I. 447. A most megjelenő **C. 1460.** feladatban leírt hópehely grafikus megvalósításával készítsünk hóesés szimulációt. A havazás egy percig tartson úgy, hogy kezdetben kevés, majd egyre több, végül ismét kevés hópehely jelenjen meg a képtér felső részén. A hópelyhek mérete változatos legyen, de a legnagyobb és legkisebb átmérőjének aránya ne érje el a kettőt. A hópelyhek a képtér tetején vegyesen, a képződés első vagy második lépése utáni állapotban hulljanak lefelé. A hópelyhek esési sebességét a méretükkel egyenesen arányosra állítsuk: a legkisebbek essenek a legnagyobb sebességgel. A képtér alján a hópelyhek tűnjenek el.

A feladat megoldásához a versenykiírásban szereplő programozási nyelvek mellett HTML/CSS/Javascript segítségével készült megoldásokat is elfogadunk. A versenykiírás szerinti programozási nyelveken készült megoldások legyenek fordíthatók és futtathatók kiegészítő grafikus könyvtárak nélkül, tehát csak a fejlesztői környezetben megtalálható, beépített grafikai modulok fölhasználásával. A böngészőben futó megoldásoknak offline is működniük kell, azaz nem tartalmazhatnak külső hivatkozásokat. Javasoljuk például a HTML5 Canvas használatát.

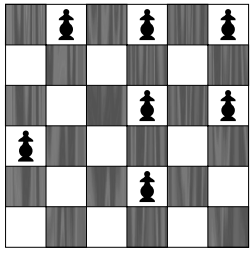
Beküldendő egy `i447.zip` tömörített állományban a program forráskódja, továbbá egy rövid dokumentáció, amely tartalmazza, hogy a forrásállomány melyik fejlesztői környezetben fordítható.

I/S. 23. Egy $N \times N$-es „sakktábla" világos színű mezőin S számú sötét gyalogot helyezünk el. Feladatunk az, hogy egy világos futó segítségével közülük minél többet leüssünk. A futó a szokásos módon mozoghat a táblán, de irányt váltani csak akkor tud, ha olyan mezőre lép, amelyen éppen leüt egy gyalogot. A futó kezdeti helye nincs rögzítve, azt mi választhatjuk meg.

Készítsünk programot, amely megadja, hogy adott állások esetén legföljebb hány gyalogot lehet a tábláról levenni.

A program standard bemenete az N és S egészek, valamint a következő S sor mindegyikében egy egész számpár. A számpárok a sötét gyalogok helyzetét adják meg: a bal felső (sötét) mezőtől indulva az egyes gyalogok sorát és oszlopát. A program standard kimenete legyen a levehető gyalogok maximális száma.

Példa bemenet (az újsor karaktereket / jelöli)	Kimenet
6 7 / 1 2 / 1 4 / 1 6 / 4 1 / 3 4 / 5 4 / 3 6 /	4



Magyarázat: a 4 1, 1 4, 3 6 és 5 4 mezőkön lévő gyalogok leütése pl. ebben a sorrendben lehetséges, ha a futó a 4 1 mezőről indul, de sajnos a többi gyaloghoz a futó nem tud eljutni.

Korlátok: $4 \leqslant N \leqslant 100$, $4 \leqslant S \leqslant 2 \cdot N$.

Értékelés: a megoldás lényegét leíró dokumentáció 1 pontot ér. További 9 pont kapható arra a programra, amely a korlátoknak megfelelő bemenetekre helyes kimenetet ad 1 másodperc futásidő alatt. Részpontszám kapható arra a programra, amely csak kisebb N értékek esetén ad helyes eredményt 1 másodpercen belül.

Beküldendő egy `is23.zip` tömörített állományban a megoldást leíró dokumentáció és a program forráskódja.

(*Az októberi számunkban megjelent* **K. 513.** *feladat alapján*)

S. 122. Legyen A az első P prímszám halmaza, és B egy N elemű, pozitív egészeket tartalmazó halmaz. Készítsünk 1-től kiindulva egy sorozatot, amelyben a sorozat következő tagja az előzőnek egy A-beli prímmel vett szorzata. Feladatunk az, hogy úgy képezzük a sorozat tagjait, hogy abban a lehető legtöbb B-beli szám forduljon elő.

Készítsünk programot, amely megadja, hogy adott A és B halmaz esetén mennyi a legtöbb olyan B-beli szám, amely egy szorzással keletkező sorozat tagjaként a fenti módon előállítható. A program standard bemenete P és N, valamint a következő N sor mindegyikében egy pozitív egész szám a B halmazból. A program standard kimenete a képzett sorozatban előforduló B-beli számok maximális száma.

Példa bemenet (az újsor karaktereket / jelöli)	Kimenet
3 10 / 5 / 6 / 8 / 10 / 9 / 12 / 21 / 16 / 18 / 24 /	3

Korlátok: $2 \leqslant P \leqslant 100$, $2 \leqslant N \leqslant 10^6$, a B halmaz minden eleme $\leqslant 10^9$.

Értékelés: a megoldás lényegét leíró dokumentáció 1 pontot ér. További 9 pont kapható arra a programra, amely a korlátoknak megfelelő bemenetekre helyes kimenetet ad 1 másodperc futásidő alatt. Részpontszám kapható arra a programra, amely csak kisebb P és N érték esetén ad helyes eredményt 1 másodpercen belül.

Beküldendő egy `s122.zip` tömörített állományban a megoldást leíró dokumentáció és a program forráskódja.

❄

A feladatok megoldásai regisztráció után a következő címen tölthetők fel:
`https://www.komal.hu/munkafuzet`
Beküldési határidő: 2018. február 10.