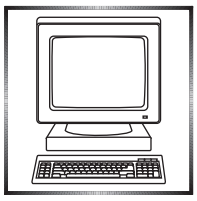


Informatikából kitűzött feladatok

I. 520. A prímszámok híres és jól ismert egészek. Néha azonban ők is szívesen elrejtőznek. Ilyenkor belebújnak egy összetett szám ruhájába. Ez úgy lehetséges, hogy a számjegyeik helyét egy vagy több forgatással megváltoztatják. A forgatás azt jelenti, hogy a szám utolsó számjegye átkerül a szám elejére. Például a 347 forgatásai a 734 és a 473. A 347 prím, de a két elforgatása összetett szám, ezért mindkettő lehet a 347 álruhája, tehát ez egy olyan prím, ami el tud rejtőzni.

Mivel a számok elején a vezető 0-kat nem írjuk ki, ezért a 107 forgatásának csak a 710-et tekintjük, a 71-et nem. Ha egy prím minden elforgatottja prím, akkor egyikük sem tud elrejtőzni. Ha egy szám és minden elforgatottja összetett, akkor ők nem lehetnek egy prím álruhái.

Készítsünk programot, amely a legföljebb négyjegyű pozitív egészek között megkeresi azokat, amelyek a fent leírt módon elrejthetnek egy prímet.

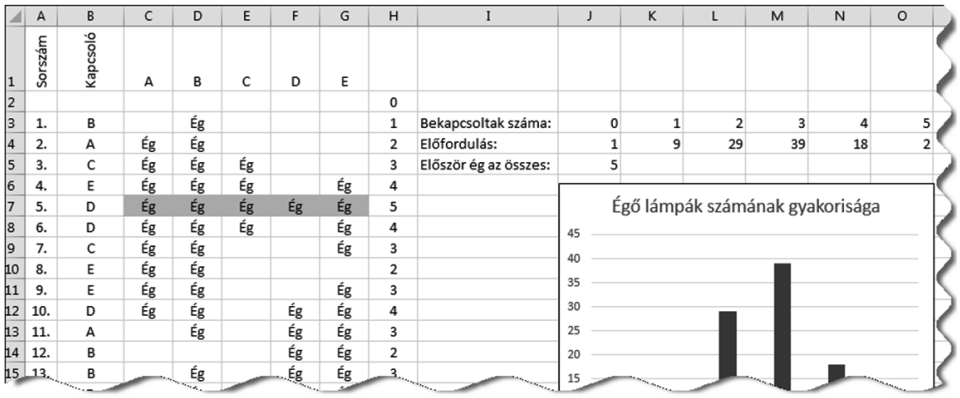
A program a standard kimenet első sorába írja ki az elrejtésre alkalmas egészek számát, második sorába növekvő sorrendben, vesszővel elválasztva az elbújtatásra alkalmas egészeket.

Beküldendő egy `i520.zip` tömörített állományban a forrásprogram és egy rövid dokumentáció, amely megadja, hogy a program melyik fejlesztői környezetben futtatható.

I. 521. Egy lapra A, B, C, D és E jelöléssel 5 darab lámpa van egy sorban elhelyezve. A lámpák függetlenek egymástól, mindegyikhez egy-egy kapcsoló tartozik, amellyel csak a hozzá tartozó lámpa be- és kikapcsolása végezhető. Kezdetben minden kapcsoló kikapcsolt állapotban van. A kapcsolókat véletlenszerűen kapcsoljuk egyik állásból a másikba, és figyeljük az égő lámpák számát, majd kellő számú kísérlet után ezen számok gyakoriságát.

A kísérletet szimuláljuk és értékeljük ki táblázatkezelővel.

1. Hozzunk létre egy munkafüzetet `i521` néven és mentsük a táblázatkezelő alapértelmezett formátumában, majd abban nevezzünk el egy munkalapot **lampak** néven.
2. Előkészítésként adjuk meg a táblázat 1. sorát és A oszlopát a *minta* szerint. A feliratokkal a szimuláció értelmezését segítjük.
3. Az A oszlopban a kapcsolók megnyomásának sorszámát, míg a B oszlopban a véletlenszerűen választott kapcsoló betűjelét jelenítsük meg függvény segítségével. A szimuláció 100 kapcsolást tartalmazzon.
4. A C:G oszlopokban az égők be-, illetve kikapcsolt állapotát jelenítsük meg Ég felirattal, illetve üres cellával.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Sorszám	Kapcsoló	A	B	C	D	E								
2								0							
3	1.	B		Ég				1	Bekapcsoltak száma:	0	1	2	3	4	5
4	2.	A	Ég	Ég				2	Előfordulás:	1	9	29	39	18	2
5	3.	C	Ég	Ég	Ég			3	Először ég az összes:	5					
6	4.	E	Ég	Ég	Ég		Ég	4							
7	5.	D	Ég	Ég	Ég	Ég	Ég	5							
8	6.	D	Ég	Ég	Ég		Ég	4							
9	7.	C	Ég	Ég			Ég	3							
10	8.	E	Ég	Ég				2							
11	9.	E	Ég	Ég			Ég	3							
12	10.	D	Ég	Ég		Ég	Ég	4							
13	11.	A		Ég		Ég	Ég	3							
14	12.	B				Ég	Ég	2							
15	13.	B		Ég		Ég	Ég	3							

5. A H2:H100 cellákban minden kapcsolás után írjuk ki, hogy aktuálisan hány lámpa ég.
6. Feltételes formázással emeljük ki azokat a sorokat, ahol mind az öt lámpa ég (lásd a mintát). Az J6-os cellában írjuk ki, hogy hányadik kapcsolásnál fordult először elő az öt lámpa egyidejű bekapcsolt állapota.
7. Az I3:I4 cellák feliratát készítsük el és mellette a J3:G4 cellákban határozzuk meg másolható függvény segítségével az égő lámpák számának gyakoriságát.
8. A gyakoriságot feliratokkal ellátott diagramon ábrázoljuk a minta szerint a felette lévő adatok oszlopainak szélességében.
9. Az A:H oszlopok tartalmát igazítsuk középre. Az A1:B1 cellákban az írásirányt állítsuk függőlegesre.

Beküldendő egy `i521.zip` tömörített állományban a megoldást adó munkafüzet és egy rövid dokumentáció, amelyből kiderül az alkalmazott táblázatkezelő neve és verziószáma.

I. 522. A Very Hard Fegyintézet fegyőrei és rabjai kizárólag egyetlen adat alapján ítélik meg magukat és egymást: kinek mekkora a bicepsze. Adataikat a `fegyorok.txt` és a `rabok.txt` szöveges állomány tartalmazza: egy szóközzel elválasztva előbb a fegyőrök (illetve rabok) azonosítója, majd bicepszének kerülete szerepel centiméterben kifejezve. A fegyőrök azonosítója háromjegyű egész szám, amely előtt egy R betű van, míg a rabok azonosítója egy négyjegyű egész szám. Például a `fegyorok.txt` állományban az R162-es azonosítójú fegyőr bicepsze 50,2 cm:

```
R162 50,2
R176 21,5
```

míg a rabok.txt állományban:

```
1717 26,6
2563 40,5
```

a 2563-as rabé 40,5 cm. A fegyházban legfeljebb 40 fegyőr, és legfeljebb 100 rab van.

1. Olvassuk be és tároljuk el az adatokat két adatállományból.

2. Hány fegyőr, és hány rab van az intézetben? Írassuk ki a képernyőre a létszámokat.

3. Kérjük be egy rab azonosítóját, majd írassuk ki bicepszének méretét. Ha nincs ilyen azonosítójú rab, akkor jelenjen meg egy erre utaló üzenet.

A fegyházban a rabok egyenként sétálnak. Minden rabot egy sétára fegyőrnek kell kísérni, de olyannak, akinek a bicepsze nagyobb, mint az adott rabé.

4. Kérjük be egy fegyőr azonosítóját (feltehetjük, hogy van ilyen), és írassuk ki, hogy hány olyan rab van, akit elkísérhet sétálni.

5. Van-e olyan rab, aki sohase mehet sétálni? Ha igen, írassuk ki a képernyőre az azonosítóját! Ha nincs, akkor írassuk ki: „Minden rab mehet levegőzni!”

6. A rabok titokban „Szökéselőkészítő Tanácsot” alakítanak. A tanácsnak a három legnagyobb bicepszű rab lesz a tagja. Írassuk a tanács tagjainak azonosítóját a titok.txt szöveges állományba (feltehetjük, hogy nincsenek azonos bicepszű rabok).

Beküldendő egy `i522.zip` tömörített állományban a megfelelően dokumentált és kommentezett forrásprogram, amely tartalmazza a megoldás lépéseit, valamint megadja, hogy a program melyik fejlesztői környezetben futtatható.

I/S. 48. Egy sakktáblára szeretnénk minél több vezért elhelyezni úgy, hogy egyik se üsse a másikat. Valaki már el is helyezett K vezért a táblára úgy, hogy egyik se üti a másikat. Adjuk meg, hogy legfeljebb hány vezért helyezhetünk még el a táblára úgy, hogy a táblán levő vezérek közül egyik se üsse a másikat.

Bemenet: az első sor tartalmazza a K számot. A következő K sor mindegyike egy x és egy y számot tartalmaz: az adott vezért az x-edik sor y-adik oszlopába tették le.

Kimenet: adjuk meg, hogy legfeljebb még hány vezért tehetünk a táblára úgy, hogy egyik se üsse a másikat.

Példa:

Bemenet (a / jel sortörést helyettesít)	**Kimenet**
3 / 2 7 / 6 2 / 7 6	5

Korlátok: $1 \leqslant K \leqslant 7$, $1 \leqslant x, y \leqslant 8$. *Időkorlát:* 0,3 mp.

Értékelés: a pontok 50%-a kapható, ha $K = 7$.

Beküldendő egy `is48.zip` tömörített állományban a megfelelően dokumentált és kommentezett forrásprogram, amely tartalmazza a megoldás lépéseit, valamint megadja, hogy a program melyik fejlesztői környezetben futtatható.

S. 147. Ha egy számítógépes rendszerben egyszerre több program fut, akkor zárakat használnak a megosztott erőforrások biztonságos kezelése érdekében. Mielőtt egy program pl. közös memóriaterületet használna, megvizsgálja, hogy a hozzá tartozó zár nyitva van-e. Ha igen, akkor zárja azt, használja az erőforrást, majd ha már nincs rá szüksége, akkor a zárat kinyitja. Amíg a program használja az erőforrást és az ahhoz tartozó zár be van zárva, addig más program nem tudja az erőfor-

rást használni, így várakoznia kell, amíg az erőforrás szabaddá nem válik és a zár újra nyitva nem lesz.

Van két programunk, melyek önmagukban determinisztikusak, azaz véges időn belül lefutnak és pontosan tudjuk, hogy az egyes utasításaikat milyen sorrendben hajtják végre. Mindkét program esetében ismerjük, hogy mely erőforrásokat és milyen sorrendben próbálják lefoglalni és elengedni, tehát azok zárjait milyen sorrendben nyitják és zárják.

Döntsük el, hogy kialakulhat-e holtpont, ha csak ez a két program fut a rendszerben. Holtpontnak nevezzük azt az állapotot, amikor a rendszerben futó összes program olyan erőforrásra vár, amelyet egy másik program már használ, ezért egyik program sem tud tovább futni. Készítsünk programot, amely T programpár esetében meghatározza, hogy kialakulhat-e holtpont.

Bemenet: az első sor tartalmazza a programpárok T számát. Minden programpárt három sor ír le. Az első sor tartalmazza az N és M számokat, melyek az első és a második program zár műveleteinek számát adja meg. A következő két sor az első és a második program erőforrás műveleteit írja le a programfutás szerinti sorrendben. Egy $x > 0$ szám azt jelenti, hogy a program a következő lépésében az x-edik erőforrást használná, míg egy $x < 0$ szám azt, hogy az x-edik erőforrást a program már nem használja tovább. Ha a program futása végére ér, akkor elengedi az összes lefoglalt erőforrást, tehát minden általa lezárt zár kinyílik. Ez nem feltétlenül jelenik meg a bemenetben.

Kimenet: T sort kell kiírni, amelyek mindegyike az „`Igen`” vagy a „`Nem`” szöveg aszerint, hogy kerülhet-e holtpontba a rendszer.

Példa:

Bemenet (a `/` jel sortörést helyettesít)	**Kimenet**

`2`
`6 3 / 1 2 3 -3 -2 -1 / 1 3 2`
`4 5 / 1 2 -2 3 / 2 3 -2 -3 1`
	`Igen`
	`Nem`

Korlátok: $1 \leqslant T \leqslant 10$, $1 \leqslant N, M \leqslant 100$, $1 \leqslant x \leqslant 10^6$. *Időkorlát:* 0,2 mp.

Beküldendő egy `s147.zip` tömörített állományban a megfelelően dokumentált és kommentezett forrásprogram, amely tartalmazza a megoldás lépéseit, valamint megadja, hogy a program melyik fejlesztői környezetben futtatható.

Javasolta: *Erben Péter* és *Darabos Dániel*

❄

A feladatok megoldásai regisztráció után a következő címen tölthetők fel:
`https://www.komal.hu/munkafuzet`
Beküldési határidő: 2020. december 15.

❄