1992

1. feladat: (25 pont)

Adott egy A N(>1) elemű, nullákból és egyesekből álló sorozat, és egy $1<K\leq N/2$ egész szám. Készíts algoritmust, ami meghatározza azt a K hosszúságú sorozatot amelyik legalább kétszer előfordul a sorozatban úgy, hogy nincs közös indexű elemük! Megoldásként adjuk meg az ismétlődő sorozat első elemének indexét!

Példa:

N:=9

K:=3

A:=(1 1 1 1 1 0 1 1 0)

Itt rossz megoldás az **(1 1 1)** sorozat, mert a bármelyik három egyest vesszük az első ötből, a két sorozatnak van azonos indexű eleme. Ha az első hármat, akkor az egyik sorozat az A(**1**), A(2), A(3) a másik legjobb esetben az A(3), A(4), A(5) sorozat lehet, vagyis a 3-as indexű elem mindegyikben szerepel.

Jó megoldás az (1 1 0) sorozat, mivel az A(4), A(5), A(6) sorozat ugyanaz mint az A(7), A(8), A(9) sorozat **é**s nincs közös indexű elemük**!**

2. feladat (75 pont)

Olvass be egy aritmetikai kifejezést, bontsd lexikális elemekre, majd ellenőrizd a helyességét!

Az alábbi példa ezeknek a fázisoknak egyszerű bemutatására szolgál:

A kiértékelés menete:

A kiértékelendő kifejezés: 3*4/(52+6*sin(7))

Lexikális analízis: 3 * 4 / (52 + 6 * sin (7))

Szintaktikai ellenőrző: A kifejezésben nincs hiba.

A kifejezésben előfordulhatnak fixpontos valós számok, X és Y paraméterek.

A következő csoport, amelyikkel foglalkozni kell, az **alapoperátorok** társasága. Az unáris mínuszt az eredményben meg kell különböztetni a kivonás műveletétől

Alapoperátorok:

–	←	kivonás vagy unáris minusz
+	←	összeadás
*	←	szorzás
/	←	osztás
^	←	hatványozás

Lehessen **zárójelek**kel a műveletek feldolgozási sorrendjét befolyásolni! A kifejezés tartalmazhat függvényeket:

Az általunk használt függvények:			
ABS	EXP	LN	SQR
ARCTAN	FRAC	ROUND	SQRT
COS	INT	TRUNC	SIN

A részekre bontás után ellenőrizni kell a kifejezés helyességét.

Ezt a műveletet két részre bonthatjuk. Egyik részfeladata megállapítani a tagokról, hogy önmagukban helyesek-e (pl. 3.3.3 nem jó szám!), másik rész azt mondja meg, hogy az egyes tagok jó helyen állnak-e (pl. a*/b, sin(), (a–b)) kifejezések rosszak).

Tagok ellenőrzésekor egy szám rossz, ha két tizedespont van benne, jel rossz, ha nem szerepelhet a kifejezésben, szöveg rossz, ha nem paraméter neve és nincs felsorolva a függvények között.

Az elemek elhelyezkedésének vizsgálatához megadjuk, hogy mi miután nem következhet, mikor van valami rossz helyen:

– **numerikus konstans** rossz helyen áll, ha:

  – paraméter,
  – függvény (nyitózárójel nélküli), vagy
  – csukózárójel van előtte.

– **operátor:**

  Háromféle operátor szerepelhet a kifejezésben

  a) kétoperandusú művelet (+ * / ^) nem jó helyen áll, ha

    – a legelső, vagy legutolsó tagja a kifejezésnek,
    – előtte operátor, vagy
    – nyitózárójel van;

  b) unáris mínusz, kivonás nincs jó helyen, ha

    – az előző tag is - jel,
    – függvényhívás volt az előző rész, vagy ha
    – utolsóként szerepel a kifejezésben;

  c) függvényhívást hibásan alkalmazzuk, ha

    – ő az utolsó rész a tagok sorában,
    – nem követi nyitózárójel,
    – előtte nem nyitózárójel, nem kétoperandusú művelet,
    – vagy nem előjelváltás áll.

– **paraméter** hibás ha**:**

  – előtte szám van, vagy
  – közvetlenül csukózárójel mögött áll.

– **zárójelek** rosszak ha:

  – nincsenek megfelelően párban,
  – egy nyitó előtt rögtön szám, vagy paraméter áll, ha előtte csukózárójel van,
  – a csukózárójel nyitó után következik, vagy ha előtte valamilyen operátor van.

1993

1. feladat (20 pont)

Morze-jelekel a magyar ábécé betűi egyértelműen kódolhatók a következő táblázat alapján (- jelöli a hosszú, . a rövid jelet):

a : .-	é : ..-..	k : -.-	p : .--.	ü : ..--
á : .---	f : ..-.	l : .-..	q : --.-	v : ...-
b : -...	g : --.	m : --	r : .-.	w : .--
c : -.-.	h :	n : -.	s : ...	x : -..-
d : -..	i : ..	o : ---	t : -	y : -.--
e : .	j : .---	ö : ---.	u : ..-	z : --..

Az egyes betűk jelei után egy szóköz, a szavak között két szóköz van.

Készíts programot, amely egy karakterekkel felírt szöveget Morze-jelekké alakít, Morze-jelekkel felírt szöveget pedig karakterekkel felírttá!

2. feladat (30 pont)

Ismerjük a síkon N db pont koordinátáit. Készíts programot, amely kirajzolja a pontokat, majd meghatározza és kirajzolja azt a minimális méretű konvex sokszöget, amely az összes pontot belső- vagy határpontként tartalmazza! (Segítségül az elinduláshoz: érdemes keresni 4 "szélső helyzetű" pontot, ezek biztosan pontjai lesznek a sokszögnek. Ezután további pontokkal kell "csak" bővíteni...)

1994

1. feladat: (25 pont)

Egész számokat a 10-es mellett -10-es számrendszerben is felírhatunk, a következőképpen:

$X=X_0+(-10)*X_1+(-10)^2*X_2+...+(-10)^n*X_n$,

ahol $0 \le X_i \le 9$. Ebben a számrendszerben minden szám felírható előjelnélküli számként. Készíts programot, amely egy maximum négyjegyű 10-es számrendszerbeli számot -10-es számrendszerbelivé vált, illetve fordírtva: -10-esből 10-esbe!

Példa:

10-es	-10-es	számrendszer
3	3	
-6	14	(=-10+4)
34	174	(=100-70+4)
-72	88	(=-80+8)
163	243	(=200-40+3)
-527	1533	(=-1000+500-30+3)
1526	19686	(=10000-9000+600-80+6)
1994	18014	(=10000-8000+0-10+4)

2. feladat: (25 pont)

Bernoulli folyadékok keveredésére a következő modellt találta ki: vegyünk két dobozt, véletlenszerűen teletöltve a két folyadékot jellemző A és B betűkkel:

1. doboz:	A	A	A	B	A	B
2. doboz:	B	A	A	B	B	A

A szimuláció egy lépésében egy-egy véletlen helyet választunk mindkét dobozban, majd az ott található betűket felcseréljük egymással. A szimuláció során figyeljük, hogy hogyan alakul az első, illetve a második dobozban.

Készíts programot, amely feltölti véletlenszerűen a két dobozt, lejátsza a fent leírt szabályokkal az eseményeket, s eredményként folyamatosan kijelzi a két doboz állapotát, az egyes dobozokban levő A-betűk számát, s az eltelt szimulációs lépések számát!

A program futását a felhasználó állíthatja le egy tetszőleges billentyű lenyomásával, s ekkor a program megadja, hogy a futási idő alatt hányszor fordul elő az az eset, hogy az első dobozban pontosan 0, 1, 2, ... db A-betű volt.

1995

1. feladat: "nagyon" prímek előállítása (30 pont)

"Nagyon" prímeknek nevezzük az olyan prímszámokat, amelyek bármely kedőszelete is prímszám. Például nagyon prím a 239, mert a 2, a 23 és a 239 is prím; nem nagyon prím a 241, ami ugyan prímszám, a 2 is prím, de a 24 nem az.

Készíts programot, amely előállítja az összes n-jegyű nagyon prímet! A program írja ki ezen számok kezdőszeleteit is! A helyes megoldások közül többet ér az, amelyik rövidebb idő alatt fut le.

Példa: 2, 23, 233 2, 23, 239 2, 29, 293 3, 31, 311 ...

2. feladat: mozgás gravitációs térben (46 pont)

Két pontszerű test mozog a síkon. Mindkettőnek adott a tömege, kezdőkoordinátái és kezdősebessége. A két test egymást a gravitációs erővel vonzza, a testekre más erő nem hat.

Készíts programot, amely a következő ötlet alapján modellezi a két test mozgását! A megfigyelés idejét egyenlő részekre osztjuk. A testek sebességének és helyének Dt időegység alatti változását a következő módon számíthatjuk:

$$v = v_0 + a * \Delta t \qquad x = x_0 + v * \Delta t$$

ahol v_0 és x_0 a test előző pillanatbeli, v és x pedig az új sebesség- és helyvektora. Az a a gyorsulásvektor.

A két test között fennálló erőt a következő módon számítjuk:

$$F = k \frac{m_1 * m_2}{r^2}$$

ahol r a két test távolsága k pedig egy konstans, amit tekintsünk 1-nek.

Ebből az i. test gyorsulásvektora az

$$F = m_i * a$$

képletből számítható.

A program olvassa be a két test kezdőadatait (tömeg, hely, sebesség), majd a képernyőre rajzolja ki a testek mozgását a következő változatokban. A rajzolás módja szerint:

- mindkét testet letörli a régi helyéről és kirajzolja az új helyre;
- úgy rajzolja az új helyre a testeket, hogy a régiről nem törli (így megmarad a pálya képe);

A koordinátarendszer szempontjából:

- A (0,0) koordinátájú pont a képernyő közepe, s a két testet ehhez viszonyítva rajzoljuk;
- Az első test mindig a képernyő közepén van, s a másodikat hozzá viszonyítva rajzoljuk.

A program háromféleképpen állhat le:

- billentyűlenyomásra;
- ha a két test összeütközik;
- ha az egyik test már biztosan nem tér vissza (azaz parabola- vagy hiperbolapályán elhagyja a rendszert) – írd le, miből lehet erre rájönni!

3. feladat: bizonyítás programmal (24 pont)

Igazold programmal a következő állítást!

Képezzük az A(N) számsorozatot a következő módon: A(1) legyen tetszőleges, nemnegatív egész szám, A(I+1) legyen A(I) számjegyei négyzetösszege.

Ekkor az A(I) számsorozat valamelyik tagjától kezdődően mindig ciklikus és az ismétlődő ciklus

- vagy a 0,
- vagy az 1,
- vagy a 4, 16, 37, 58, 89, 145, 42, 20

valamelyike!

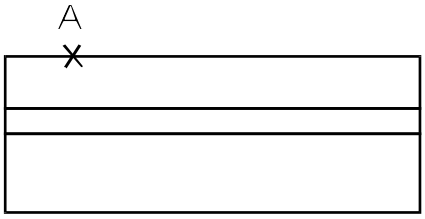
A programmal igazolást elég azon kezdőszámokra elvégezni, amelyek nem nagyobbak, mint a sorozat őket követő tagja (azaz a számjegyeik négyzetösszege).

A. Vezesd le, hogy melyik az a legnagyobb szám, amit meg kell vizsgálni (bizonyítsd be, hogy a nála nagyobbak mindegyike nagyobb, mint számjegyeik négyzetösszege)!

B. Írj programot, amely kipróbál minden, a levezetésben meghatározott számnál kisebb számot kezdőszámként, s akkor áll le, amikor valamelyik ciklust felismeri! Közben kiírja a kezdőszámot, a generált számokat, a felismert ciklust, valamint a megfigyelt lépésszámot! Ha nem tudtad megoldani az A részfeladatot, akkor olvasd be azt a számot, ameddig ki kell próbálni a sorozatok kezdőszámait!

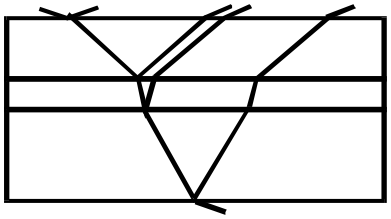
1996

1. feladat: fénytörés és visszaverődés (65 pont)



1. ábra

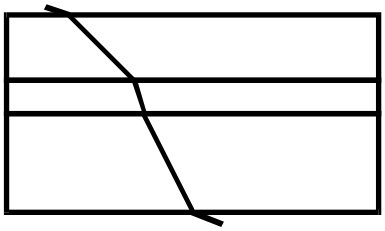
Különböző vastagságú átlátszó lemezeket helyezünk egymás mögé az 1. ábrán látható módon (3 lemez, mindkét oldalon levegő). Egy fénysugár lép be az A-val jelölt ponton (ezt tekintjük az origónak), a merőlegeshez képest alfa szöggel. Az egyes közeghatárokról a lefelé haladó fény részben visszaverődik (akkor is rajzolni kell, ha a valóságban szinte láthatatlan lenne – a fényerősséggel most nem foglalkozunk), részben pedig a fénytörés miatt a másik közegben más szögben folytatja útját (Snellius-Descartes törvény). Felfelé haladva csak a fénytöréssel kell foglalkozni, a visszaverődéssel nem.



2. ábra

Megadjuk a lapok vastagságát és –közös– szélességét, valamint az A pont távolságát a lemezek bal oldalától. Minden laphoz meg kell adni a levegőhöz viszonyított törésmutatóját. Eredményként ki kell rajzolni a lemezek oldalnézeti képét, valamint a fénysugár útját. (A rajzra példa látható a második ábrán.)

Készíts programot, amely a fenti adatokat beolvassa, majd az eredményt rajzolja és a kilépések helyét megadja (a példában fönt 4, lent 1 helyen lép ki fénysugár). A rajzot két változatban készítsd el! Az elsőben ha a fénysugár átlép a következő közegbe, akkor a visszaverődést elhanyagolhatónak tekintjük és így nem ábrázoljuk (3. ábra), a másodikban pedig az összes felfelé visszavert fénysugarat is ábrázolni kell.



3. ábra

A szükséges fizikai/matematikai ismeretek:

Snellius-Descartes törvény: $\frac{\sin(\alpha_{i-1})}{\sin(\alpha_i)} = \frac{n_i}{n_{i-1}}$, azaz $\sin(\alpha_i) = \sin(\alpha_{i-1})\frac{n_{i-1}}{n_i}$.

Ebből $\alpha_i = \arcsin\left(\sin(\alpha_{i-1})\frac{n_{i-1}}{n_i}\right)$ ahol α_i az i. közegben a merőlegessel bezárt szög, n_i pedig a közeg törésmutatója.

Az arkusz szinusz függvény kiszámítása:

$\sin(\alpha) = x$ és $\sin^2(\alpha) + \cos^2(\alpha) = 1$. Ebből következik: $\cos(\alpha) = \sqrt{1-x^2}$.

Mivel $\tan(\alpha) = \frac{\sin(\alpha)}{\cos(\alpha)}$, tehát $\tan(\alpha) = \frac{x}{\sqrt{1-x^2}}$. Ekkor $\alpha = \arctan\left(\frac{x}{\sqrt{1-x^2}}\right)$.

Tehát $\arcsin(x) = \arctan\left(\frac{x}{\sqrt{1-x^2}}\right)$.

2. feladat: Mandelbrot és Julia halmaz rajzolás (35 pont)

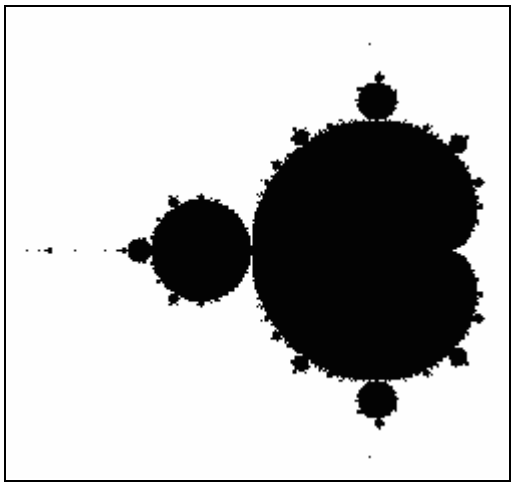
A Mandelbrot halmazba olyan síkbeli (x_0,y_0) pontok tartoznak, amelyeket kezdőértékként véve az $(x_{n+1},y_{n+1}):=(x_n*x_n-y_n*y_n+x_0,2*x_n*y_n+y_0)$ sorozat minden tagjára teljesül az $x_n*x_n+y_n*y_n<4$ reláció.

A Julia halmaz a Mandelbrot halmazhoz hasonlóan képződik: tetszőleges (p,q) ponthoz tartozó Julia halmazba azon (x_0,y_0) pontok tartoznak, amelyekre az $(x_{n+1},y_{n+1}):=(x_n*x_n-y_n*y_n+p,2*x_n*y_n+q)$ sorozat minden tagjára teljesül az $x_n*x_n+y_n*y_n<4$ reláció.

Mivel programban nem számolhatjuk ki egy végtelen sorozat minden értékét, ezért csak adott n-ig (pl. az ábráknál n=150) számolunk.

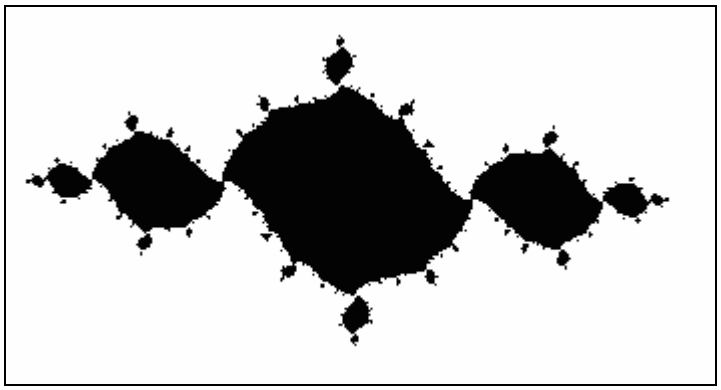
A. Állapítsd meg, hogy milyen síktartománnyal érdemes egyáltalán foglalkozni a feladat megoldásában (miket fogsz megvizsgálni a programodban)!

B. Készíts programot, amely kirajzolja a képernyőre 50-szeres nagyításban a Mandelbrot halmazba tartozó pontokat! (4. ábra)



4. ábra

C. Készítsd el egy tetszőleges beolvasott (p,q) ponthoz tartozó Julia halmaz képét. (5. ábra – a (-1,-0.1) ponthoz tartozó Julia halmaz)



5. ábra

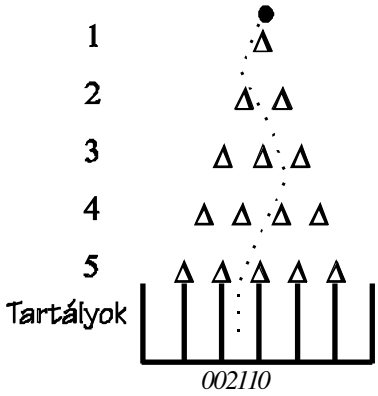
D. Bizonyítsd be, hogy ha (x,y)∈ Mandelbrot-halmaz, akkor (x,-y)∈ Mandelbrot-halmaz!

E. Bizonyítsd be, hogy ha (x,y)∈ Mandelbrot-halmaz, akkor nem biztos, hogy (-x,y)∈ Mandelbrot-halmaz! (Adj ellenpéldát, amire ezt be is látod!)

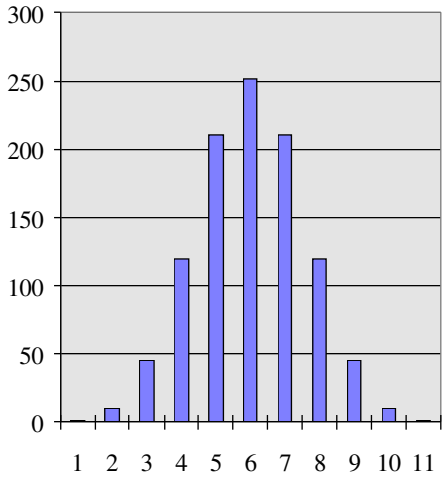
1997

1. feladat: Galton-deszka szimuláció (100 pont)

Azt, hogy a binomiális eloszlásnak „köze van” a normális eloszláshoz, a matematikusok sokféle tételben megfogalmazták. Készítsünk a két eloszlás „rokonságának” igazolására kísérleti eszközt a következő elképzelésre építve! Egy deszkába *n* sorban szabályosan elrendezve ékeket helyezünk el, a *k*. sorban éppen *k* darabot (l. az 1. ábrát). Golyókat indítunk útnak legfelül, egymás után, amelyek az ékeken véletlenszerűen eltérülve hullanak lefelé, míg végül a legalul elhelyezett *n+1* tartály valamelyikében landolnak. Az ékek 1/2 (*P*) valószínűséggel térítik el balra, illetve jobbra. Sok-sok golyóval elvégezve a kísérletet a normális eloszlás harang-görbéjére emlékeztető alakzat fog kialakulni a lehullott golyók „oszlopaiból”.



1. ábra (n=5 esetén)



2. ábra (n=10 esetén)

Ezt az ún. Galton-deszka kísérletet kell számítógéppel imitálnod. A program paraméterként kérje be a sorok számát (*n*), valamint az egyes ékekről való balra eltérülés valószínűségét (*P*). Rajzold ki a Galton-deszkát a képernyőre, majd billentyűvel vezérelhető sebességgel „kövesd” útjukon a golyókat. Az egyes tartályokba leérkező golyók száma jelenjen meg a tartályoknál, a relatív gyakorisága pedig egy hisztogramon (l. 2. ábrát), a felhasználó által kívánt golyószámonként kirajzolva. Ha nagyon gyors mozgást kíván, akkor elegendő minden –mondjuk– 100. időegységben újrarajzolni a hisztogramot, ha pedig nagyon lassan, akkor minden egyes golyó után. Mindez addig tartson, amíg a felhasználó meg nem unja.

1998

1. feladat: Faktoriális számrendszer (30 pont)

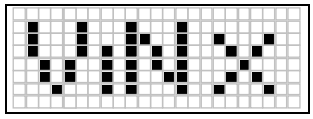
Faktoriális számrendszerben levő természetes számokat úgy kell elképzelni, hogy minden egyes helyi értéken az előző helyi értéknél eggyel nagyobb alapszámú számrendszert akalmazunk. Azaz egy ilyen szám utolsó számjegye 2-es számrendszerbeli, az azt megelőző 3-as számrendszerbeli, a még előtte levő pedig már 4-es számrendszerbeli szám, és így tovább.

Készíts programot, amely két funkcióval rendelkezik:

A. tízes számrendszerbeli szám faktoriális számrendszerre váltása, illetve

B. faktoriális számrendszerbeli szám tízes számrendszerre váltása.

2. feladat: Betűfelismerés (70 pont)



Az ábécé betűit 8xK-as pontmátrixban ábrázoljuk, 0-val kódolva a világos, és 1-gyel a sötét pontokat. A vonalak belsejében levő sötét pontoknak legfeljebb 2 sötét szomszédja lehet, a vonalak találkozásánál levőknek 3 vagy 4, a vonalak végén levőknek pedig 1. A betűket legalább egy üres oszlop választja el egymástól és különböző méretűek, esetleg elnyújtottak, de egyenes állásúak lehetnek.

A BETUKx.BE állomány első sora a szöveg szélessége (K≤100), a további 8 sor pedig az egyes betűk leírása, minden sorban K számjegy (0 vagy 1). Az állományban csak az angol ábécé nagybetűi fordulnak elő:

ABCDEFGHIJKLMNOPQRSTUVWXYZ.

Készíts programot, amely a bemeneti állományból a vonalak jellege és végpontjaik helyzete alapján felismeri az I, N, V, X betűket! A BETUKx.KI állományba és képernyőre egyetlen sort kell írni, amely a bemenő állomány sorrendjében tartalmazza a felismert betűket. A fenti 4-től különböző betűk esetén (ha a fentiek alapján megkülönböztethető tőlük – az L-től a V például biztosan nem különböztethető meg) az eredmény megfelelő helyén a – karakter szerepeljen!

Példa:

BETUK1.BE	BETUK1.KI
14	XI-
00000000000000	mert a 3. betűként a példán
00000101000001	látható Y-t a programnak nem
00000100100010	kell felismernie.
01010100010100	
00100100001000	
01010100001000	
00000100001000	
00000000000000	

Elérhető összpontszám: 100 pont

1999

1. feladat: Faktoriális prímtényezős felbontása (35 pont)

A Faktoriális(N) függvény rendkívül gyorsan növekszik. Míg az 5!=120, addig már a 10!=73628800 ábrázolásához 4 byte-os egész számokra van szükség. A 100! azonban sem 4, sem 8, … byte-os egész számként nem ábrázolható a számítógépben.

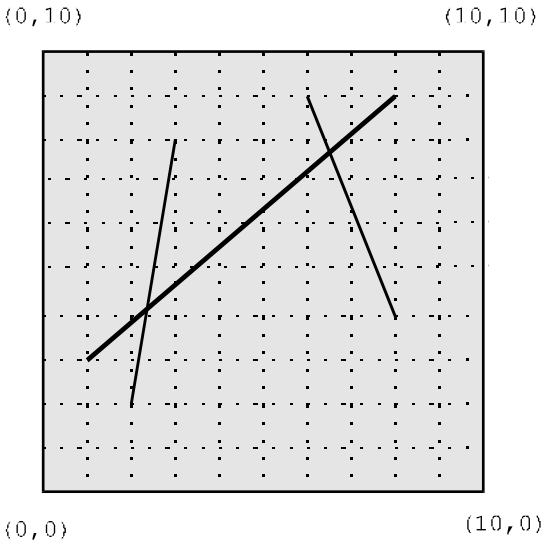
Tudjuk azonban, hogy minden természetes számnak elkészíthető a prímtényezős felbontása. Például:

$$5! = 2^3 * 3 * 5 \qquad 10! = 2^8 * 3^4 * 5^2 * 7$$

Készíts programot, amely beolvassa billentyűzetről N értékét ($1 \le N \le 10000$), majd kiírja a képernyőre az N! prímtényezős felbontását!

2. feladat: Szakaszok metszéspontjai (65 pont)

Egy téglalap alakú területre N db ($1 \le N \le$ 100) szakaszt rajzoltunk. Az egyes szakaszok végpontjainak koordinátái 0 és 100 közötti egész számok. Két szakasz akkor metszi egymást, ha pontosan egy közös pontjuk van.



Készíts programot, amely meghatározza a szakaszok metszéspontjait! Ha egy pont kettőnél több szakasznak is a metszéspontja, akkor is csak egyszer szabad feltüntetni az eredményben.

A SZAKASZ.BE állomány első sorában a szakaszok száma van. A következő N sor mindegyike 4 számot tartalmaz (x_1, y_1, x_2, y_2,), a szakasz kezdő és végpontjának egész koordinátáit, egy-egy szóközzel elválasztva.

A SZAKASZ.KI állományba és a képernyőre a szakaszok metszéspontjait kell írni: az első sorba a metszéspontok K számát, a következő K sorba pedig az egyes metszéspontok koordinátáit, 3 tizedesjegyre kerekítve.

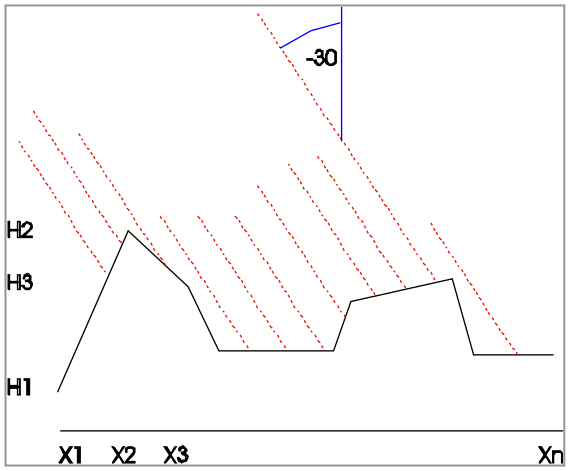
Példa:

SZAKASZ.BE:	SZAKASZ.KI:
3	2
2 2 3 8	2.361 4.167
1 3 8 9	6.239 7.537
6 9 7 4	

2000

1. feladat: Árnyékos területek (70 pont)

Egy látképet egyenes szakaszok sorozatával adunk meg. A látkép felett a függőleges iránnyal az óramutató járása szerint α szöget bezárva, végtelen távolságban van a Nap.



- Add meg, hogy a Nap megvilágítja-e a teljes látképet!
- Ha nem, akkor add meg a megvilágítás irányából az első olyan szakasz sorszámát, amelyet a Nap nem világít meg!
- Add meg az összes olyan szakasz sorszámát, amelyek teljesen árnyékban vannak, illetve amelyeknek valamely részét a Nap nem világítja meg!

Bemenet: TAJKEP.BE

1. sor: M (2≤M≤1000), α (-90<α<90): egész számok szóközzel elválasztva, a látkép töréspontjainak száma, beleértve az első és az utolsó pontot is, valamint a napsugár merőlegessel bezárt szöge.

2..*M*+1. sorok: X_i H_i : Két egész szám, egy szóközzel elválasztva: az i-edik töréspont H_i magasságban, az X_i vízszintes pozíción van, 1≤i≤M; (1≤i≤M-1 esetén teljesül $X_{i+1} > X_i$); egy egyenes szakaszt két egymás utáni pont ad meg.

Kimenet: TAJKEP.KI és a képernyő

1. sor: IGEN, ha a teljes látkép meg van világítva, NEM, ha nem.

2. sor: Az első olyan szakasz sorszáma, ami teljes egészében árnyékban van.

3. sor: Az összes olyan szakasz sorszáma növekvő sorrendben, amelyek teljes egészében árnyékban vannak.

4. sor: Az összes olyan szakasz sorszáma növekvő sorrendben, amelyek részben meg vannak világítva, részben pedig árnyékban vannak.

2. feladat: Hatványok (30 pont)

Készíts programot, amely két adott természetes szám (2≤A≤2000, 2≤B≤2000) esetén megadja növekvő sorrendben, hogy az összes i természetes számra A^i B-vel osztva milyen maradékot adhat.

Példa: Bemenet: Kimenet:

3, 7 1, 2, 3, 4, 5, 6

Ebben az esetben 3^0 mod 7=1, 3^2 mod 7=2, 3^1 mod 7=3, 3^4 mod 7=4, 3^5 mod 7=5, illetve 3^3 mod 7=6; más maradék pedig semmilyen kitevőre nem jöhet ki.

2, 7 1, 2, 4

Ebben az esetben 2^0 mod 7=1, 2^1 mod 7=2, 2^2 mod 7=4; más maradék pedig semmilyen kitevőre nem jöhet ki.

2001

1. feladat: Közelítés (58 pont)

A $\sqrt{2}$ értékének közelítésére több módszer ismert, ilyen például Héron képlete: $x_{n+1} := \frac{1}{2} * \left(x_n + \frac{2}{x_n} \right)$

Egy középkori angol matematikustól származik a Pell-egyenlet, amely kimondja, hogy a $P^2 – M*Q^2=4$ egyenletnek végtelen sok megoldása van, ha M nem négyzetszám, így M=2 esetén is.

Ha $x_n = \frac{P_n}{Q_n}$ alakú, akkor $x_{n+1} := \frac{1}{2} * \left(x_n + \frac{2}{x_n} \right) = \frac{P_n^2 - 2}{P_n * Q_n} = \frac{P_{n+1}}{Q_{n+1}}$ szintén megoldása a Pell-egyenletnek, azaz $\lim_{n \to \infty} \frac{P_n}{Q_n} = \sqrt{2}$. Ez tehát lehetőséget teremt arra, hogy egész szám-párokkal (egy racionális szám számlálója és nevezője) adjuk meg a közelítő értéket, miközben csak egész számokkal végzünk műveleteket.

Természetesen a közelítő módszer akkor lesz jó, ha nagyon sokjegyű (legfeljebb 1000) egész számokkal is tudunk műveleteket végezni. Ehhez első lépésként meg kell határoznod egy P_0 és Q_0 értéket, amik kielégítik a Pell-egyenletet.

Készíts menürendszerű programot (PELL.PAS vagy PELL.C) három funkcióra

1. Beolvas egy legfeljebb 1000 jegyű nem nulla természetes számot, értékét 1-gyel csökkenti, majd kiírja a képernyőre.
2. Beolvas kettő legfeljebb 1000 jegyű természetes számot, összeszorozza őket, majd az eredményt kiírja a képenyőre.
3. Beolvassa N értékét, majd kiszámolja a $\sqrt{2}$ értékének közelítését N-szer alkalmazva a közelítő képletet!

2. feladat: Összeadás (42 pont)

Készíts programot (AD.PAS vagy AD.C), amely képes két sokjegyű természetes számot ($A \geq B$, $A \leq 10^{100}$) számjegyenként balról jobbra összeadni. Ez azt jelenti, hogy az eredmény számjegyei is balról jobbra állnak elő, mindig abban a pillanatban, amikor már tudjuk, hogy a tőle jobbra levők miatt nem változhat meg.

A program először beolvassa A és B számjegyei számát, majd balról jobbra olvassa egyesével az azonos helyiértékű számjegyeket, s amint lehet, írja az eredmény számjegyeit.

Példa: `A=1354506, B=54493` $\Rightarrow$ `A+B=1408999`

A számjegye: 1, B számjegye: —
A számjegye: 3, B számjegye: — A+B számjegye: 1
A számjegye: 5, B számjegye: 5 A+B számjegye: 4
A számjegye: 4, B számjegye: 4 A+B számjegye: 0
A számjegye: 5, B számjegye: 4
A számjegye: 0, B számjegye: 9
A számjegye: 6, B számjegye: 3 A+B számjegye: 8
A+B számjegye: 9
A+B számjegye: 9
A+B számjegye: 9