Tartalomjegyzék

Segédanyag

A C nyelvhez

Juhász István-Kósa Márk-Pánovics János: C példatár, 2005, PANEM.
A feladatok megoldásai megtalálhatóak a http://infotech.inf.unideb.hu/konyvek/cpeldatar/ oldalon. Minden kód HTML-ben van, ezért vagy szövegfájlként kell *menteni* (save as), vagy ki kell *másolni* egy fájlba (copy-paste). Az összes forráskód tömörítve letölthet? innen.

Az Informatika 2 tárgy wiki oldala.

A Ruby nyelvhez

Az Informatika 2 tárgy wiki oldala, és az ott található linkek.

Gyakorló feladatok

C Feladatok

Az alábbi feladatok részben a példatár példáinak kis módosításai vagy pontosításai, részben a tavalyiak ismétlései, de szerepel egy-két egyéb feladat is (néhol a feladat után kerek zárójelben az ismétlend? anyag szerepel). A SIO használatának felevenítésére egy-egy feladatot ott is kit?zünk.

1. (2.3) Írjunk programot, mely egy beolvasott évszámról eldönti, hogy szök?év-e. Ha a beolvasott szám nem pozitív, írja ki, hogy nem évszám. (if, else if, else, printf, scanf, logikai m?veletek: &&, ||..., %)
2. (2.11 a) Írjunk programot, mely összeadja az egészeket 1-t?l *n*-ig az $n(n + 1) / 2$ képletet használata nélkül. (for, i++, +=)
3. (2.17 b) Írjunk nem rekurzív programot egyetlen while-ciklussal két pozitív egész szám legnagyobb közös osztójának meghatározására, azaz az Euklideszi algoritmusra. (while)
4. (2.20) Olvassunk be egy karakterláncot, és számoljuk meg a kisbet?k számát az els? olyan karakterig, amely nem az angol ábécé egy bet?je, majd írjuk ki ezt a számot. (do while, char, getchar, islower, isalpha)
5. (3.3) Írjunk olyan programot, mely egy tömbbe beolvas 10 egész számot, majd eldönti, hogy van-e köztük két olyan, amelyek szorzata 48. (egymásba ágyazott for ciklusok)
6. (3.7) Írjunk olyan függvényt, mely egy adott $c < 10\,000\,000$ pozitív egészhez meghatározza azt a legnagyobb egész *n* számot, melyre léteznek olyan *x* és *y* pozitív egész számok, hogy $xF_{n-1} + yF_n$

= *c*, ahol F_n az *n*-edik Fibonacci-szám ($F_0 = 0$, $F_1 = 1$).
7. ismétlés: tavalyi honlapról bombaz.c
8. (3.7 Nyolc király?n? probléma - jeles szint) Írjunk programot, mely egy $n \times n$ -es sakktáblán megkeresi *n* király?n? összes olyan elhelyezéseit, ahol egyik király?n? sem üti semelyik másikat, és azokat kiírja a képerny?re!
9. (13.2) Szerelvényrendezés
10. (9.1) NyargaLó feladat
11. (SET! játék) A SET! egy 81 kártyából álló játék. Minden kártyán lév? jelet 4 tulajdonság jellemez, ezek mindegyikéb?l 3-3 változat fordul el? (így a kártyák maximális száma $3^4 = 81$ lehet). A 4 tulajdonság:
    (a) a kártyán lév? figurák száma (1, 2 vagy 3);
    (b) a kártya figuráinak színe (piros, zöld, lila);
    (c) a figurák telítettsége (üres, félig kitöltött, kitöltött);
    (d) a figurák alakja (Rombusz, Téglalap, Hullám).
    Pl. az egyik kártyán van 3 piros félig kitöltött rombusz (3PFR). Azt mondjuk, hogy 3 kártya „set"http://wiki.math.bme.hu-et alkot, ha a 3 kártya minden tulajdonság szerint vagy azonos, vagy különböz?. Pl. setet alkot az 1PKR, 2PKT, 3PKH, mert szám szerint mind különböz? (123), szín szerint mind azonos (P), kitöltöttség szerint mind azonos (K), alak szerint mind különböz? (RTH). Írjunk programot, mely beolvas 3 és 81 közé es? számú kártyaleírást, és kiírja az ezekb?l kiválasztható összes setet. A helyes kártyaleírás 4 karakterb?l áll, az els? az 1,2 vagy 3 valamelyike, a második a P, Z, L, a harmadik az U, F, K, a negyedik az R, T, H karakterek valamelyike. Pl. egy lehetséges input:
    ```
    3LKT
    1PUR
    3PKR
    2PFR
    3ZKH
    ```
    amire a helyes válasz:
    ```
    1PUR 2PFR 3PKR
    3PKR 3ZKH 3LKT
    ```
    A játék leírása megtalálható a wikipédián is, de van online változata, több is, s?t beviszek egy valódi példányt, úgyhogy „él?ben"http://wiki.math.bme.hu is ki lehet próbálni.

Ruby feladatok

Megoldások a Programozás_2/2007/Ruby_megoldások lapon

fill ? sokszög kitöltése

Egy *fekete-fehér* kép egy rácsnégyzetekb?l álló téglalap, melyben minden kis négyzetnek saját színe van: fekete (1) vagy fehér (0). A téglalap szélessége és magassága is egy pozitív egész szám. A bal alsó sarok a (0,0). A kis négyzeteket más néven *képpontnak* vagy *pixelnek* nevezzük. Az *üres kép* csupa fehér képpontból áll.

Egy fekete-fehér képet az alábbi módon kell PBM-fájlba írni:

1. a "http://wiki.math.bme.huP1 "http://wiki.math.bme.hu string
2. a kép szélessége 10-es számrendszerben (pl. "http://wiki.math.bme.hu30"http://wiki.math.bme.hu)
3. szóköz ("http://wiki.math.bme.hu "http://wiki.math.bme.hu)
4. a kép magassága 10-es számrendszerben (pl. "http://wiki.math.bme.hu20"http://wiki.math.bme.hu)
5. soremelés (a "http://wiki.math.bme.hu\n"http://wiki.math.bme.hu string)
6. Minden sorra (felülr?l lefele):

7. A soron belül minden képpontra (balról jobbra): "http://wiki.math.bme.hu0"http://wiki.math.bme.hu, ha a képpont fehér, és "http://wiki.math.bme.hu1"http://wiki.math.bme.hu, ha fekete
8. soremelés (a "http://wiki.math.bme.hu\n"http://wiki.math.bme.hu string)

Megjegyzés: Ha egy, a fenti tartalmú fájlnak .pbm kiterjesztést adunk, akkor a benne lev? képet (Linux alatt) sok képnézeget? programmal meg tudjuk nézni (Windows alatt is vannak ilyenek), például a GQview programmal így: gqview fájlnév.pbm

Megjegyzés: A PBM fájlformátum a fentinél kicsit b?vebb szintaxist is megenged, ezek a b?vítmények a feladat megoldásához nem szükségesek.

Egy útvonal (angolul path) képpontok listája. Az útvonal azt a sokszöget jelöli ki a síkon, melynek csúcsai a listában szerepl? képpontok a listabeli sorrendben. Élnek tekintjük a lista két szomszédos eleme közti szakaszt, továbbá az els? és utolsó csúcspontja közötti szakaszt. A lista legalább három csúcspontból áll. Csak konvex sokszögek megengedettek, és az élek meredeksége 45 foknak többszöröse kell legyen.

Egy útvonalat konvexnek nevezünk, ha bármely két szomszédos élére igaz, hogy az általuk bezárt el?jeles szög 0-nál nagyobb és 180 foknál kisebb; vagy bármely két szomszédos élére igaz, hogy az általuk bezárt el?jeles szög 0-nál kisebb és -180 foknál nagyobb. Szomszédosnak tekintünk két élt, ha van közös csúcspontjuk. (A konvexség fenti definíciójának intuitív magyarázata: egy útvonal konvex, ha körbemenve rajta mindig balra vagy mindig jobbra kell fordulni.)

Egy útvonal kitöltött képének azt a fekete konvex sokszöget nevezzük (bels? pontokkal együtt), melyet az útvonal határol. Pontosabban, egy útvonal kitöltött képének kirajzolásakor az alábbi képpontokat kell befeketíteni:

1. az útvonal csúcspontjai;

2. az útvonal élei, mint szakaszok képpontjai (hogy pontosan mely képpontok érintettek, az egyértelm?, mivel a szakaszmeredekség 45 fok többszöröse);

3. az összes olyan képpontot, amely az el?z? pontban felrajzolt szakaszok által definiált konvex sokszög belsejében található.

A feladat:
Írjon egy programott Ruby nyelven, amely a bemenetr?l beolvasott csúcspontok által definiált sokszöget kitöltve kirajzolja, és az eredményt PBM formátumban kiírja a fenti módon. (Könnyítés: els? lépésként olyan program írása, mely a sokszöget kirajzolja, de nem tölti ki, és esetleg a konkávot is megengedi.)

A program bemenetének els? sorában két egész szám található szóközzel elválasztva. Az els? a kép szélessége (w), a másik a kép magassága (h). A kép kezdetben teljesen fehér.

A bemenet további sorai (legalább 3 db) egy konvex útvonal csúcspontjait tartalmazzák sorrendben. Minden sor egy x koordinátát, egy szóközt és egy y koordinátát tartalmaz. A sorokat egy "http://wiki.math.bme.hu-1 -1\n"http://wiki.math.bme.hu sor zárja. Teljesül, hogy 0 ? x < w és 0 ? y < h.

Ha a bemeneten megadott útvonal élei között van olyan, melynek meredeksége nem 45 fok többszöröse, akkor a program "http://wiki.math.bme.huANGLE\n"http://wiki.math.bme.hu-t ír ki. Ellenkez? esetben, ha a bemenetben megadott útvonal nem konvex sokszöget határoz meg, akkor a program "http://wiki.math.bme.huNOT_CONVEX\n"http://wiki.math.bme.hu-et ír ki. Ellenkez? esetben a program a megadott szélesség? és magasságú fehér képre feketével felrajzolja a bemeneten megadott útvonal kitöltött képét, és a kapott képet a fenti módon PBM formátumban kiírja.

A program ezután új képpel folytatja a bemenet beolvasását mindaddig, amíg nincs vége.
Mintabemenet

```
4 4
0 0
3 0
0 3
-1 -1
4 3
0 0
3 0
0 2
-1 -1
11 5
1 4
9 4
9 1
1 1
-1 -1
```

Mintakimenet

```
P1 4 4
1000
1100
1110
1111
ANGLE
P1 11 5
01111111110
01111111110
01111111110
01111111110
00000000000
```

drawone ? rajzolóprogram ? a Draw45 rajzolási modell

A Draw45 rajzolási modell pixeles képek el?állítására használható. A modellben a rajzolóprogram a bemenetér?l beolvassa, majd végrehajtja a rajzoló utasításokat, és végül kiírja a kimenetére az eredményül kapott képet.

A pixeles kép egy rácsnégyzetekb?l álló téglalap, melyben minden kis négyzetnek saját színe van. A téglalap szélessége és magassága is egy pozitív egész szám. A bal alsó sarok a (0,0). A kis négyzeteket más néven képpontnak vagy pixelnek nevezzük.

Egy szín három komponensb?l áll: piros, kék és zöld. Minden komponens egy [0,255]-beli egész szám. A színt az egyes komponensek összeadó színkeverés szerinti egyesítésével kell megjeleníteni, például a (255,127,0) szín valahol a piros és a narancssárga között van, mert sok piros és kevés zöld keveréke. (További részletek a színek ily módon történ? összeállításáról az RGB Wikipedia-lapon.)

Egy képet az alábbi módon kell fájlba írni:

1. a "http://wiki.math.bme.huP6 "http://wiki.math.bme.hu string
2. a kép szélessége 10-es számrendszerben (pl. "http://wiki.math.bme.hu30"http://wiki.math.bme.hu)
3. szóköz ("http://wiki.math.bme.hu "http://wiki.math.bme.hu)
4. a kép magassága 10-es számrendszerben (pl. "http://wiki.math.bme.hu20"http://wiki.math.bme.hu)
5. a "http://wiki.math.bme.hu 255\n"http://wiki.math.bme.hu string
6. Minden képpontra (felülr?l lefele, azon belül balról jobbra):
    1. egy bájt, ami a képpont színének piros komponense (pl. Ruby-ban 255.chr)
    2. egy bájt, ami a képpont színének kék komponense (pl. Ruby-ban 127.chr)
    3. egy bájt, ami a képpont színének zöld komponense (pl. Ruby-ban 0.chr)

Megjegyzés: Ha egy, a fenti tartalmú fájlnak .ppm kiterjesztést adunk, akkor a benne lev? képet a legtöbb képnézeget? programmal meg tudjuk nézni, például a GQview programmal így: `gqview fájlnév.ppm`

Egy útvonal (angolul path) képpontok listája. Az útvonal azt a sokszöget jelöli ki a síkon, melynek csúcsai a listában szerepl? képpontok a listabeli sorrendben. Csak konvex sokszögek megengedettek, és az élek meredeksége 45 foknak többszöröse kell legyen.

Egy útvonalat konvexnek nevezünk, ha bármely két szomszédos élére igaz, hogy az általuk bezárt el?jeles szög 0-nál nagyobb és 180 foknál kisebb; vagy bármely két szomszédos élére igaz, hogy az általuk bezárt el?jeles szög 0-nál kisebb és -180 foknál nagyobb. Élnek tekintjük a lista els? és utolsó csúcspontja közötti élt is. Szomszédosnak tekintünk két élt, ha van közös csúcspontjuk. (A konvexség fenti definíciójának intuitív magyarázata: egy útvonal konvex, ha körbemenve rajta mindig balra vagy mindig jobbra kell fordulni.)

Egy alakzat (angolul shape) az alábbi tulajdonságok összessége: útvonal, szín, kitöltöttség. A kitöltöttség lehet üres vagy teli.

Az alakzatot az alábbi módon kell egy képre felrajzolni:

1. Az alakzat útvonalának csúcspontjait az alakzat színére kell festeni.
2. Az alakzat útvonalában szerepl? összes szomszédos csúcspont-pár által definiált szakasz minden képpontját az alakzat színére kell festeni. Hogy pontosan mely képpontok érintettek, az egyértelm?, mivel a szakaszmeredekség 45 fok többszöröse.
3. Ha az alakzat kitöltöttsége teli, akkor az összes olyan képpontot az alakzat színére kell festeni, amely az el?z? pontban felrajzolt szakaszok által definiált konvex sokszög belsejében található.
4. A rajzolóprogram egy fehér kezd?képb?l indulva beolvassa és végrehajta a bemenetén található utasításokat, majd kiírja az eredményül kapott képet a fent definiált formátumban. Ha a bemenet hibás (pl. értelmetlen utasítás szerepel rajta, vagy nem konvex útvonalat kéne definiálni), akkor az els? hibánál a program hibaüzenetet ír ki a kimenetére, majd befejezi futását (tehát az eredmény-képet nem írja ki).

Az utasítások az alábbiak:

- #akármi, ahol akármi tetsz?leges nemüres string (a sor végéig tart). Ez az utasítás csak egy megjegyzés a bemeneten, nem csinál semmit.
- CANVAS w h, ahol w és h pozitív egész számok. Az aktuális képet új, fehér (csupa (255,255,255) szín?) képre cseréli, melynek szélessége w, magassága h.
- PATH p x1 y1 x2 y2 x3 y3 ..., ahol p az angol ábécé kisbet?ib?l álló, nemüres string, x1 stb. egész szám és ... további páros sok egész szám, szóközzel elválasztva. Egy p nev? útvonalat definiál, melynek csúcspontjai (x1,y1), (x2,y2), (x3,y3) stb. Csak konvex sokszög megengedett.
- RGB c r g b, ahol c az angol ábécé kisbet?ib?l álló, nemüres string; r, g és b pedig [0,255]-beli egész számok. Egy c nev? színt definiál, amely a megadott összetev?kb?l áll.

- SHAPE s p c f, ahol s, p és c az angol ábécé kisbet?ib?l álló, nemüres string; f pedig vagy 0, vagy 1. Egy s nev? alakzatot definiál, amelynek útvonala p, színe c és kitöltöttsége f (0 jelöli az üreset, 1 a telit).
- DRAW s x y, ahol s az angol ábécé kisbet?ib?l álló, nemüres string; x és y egész számok. Az aktuális képre kirajzolja az s nev? alakzatot (x,y)-nal eltolva (vagyis minden csúcspont koordinátáihoz x-et ileltve y-t hozzáadva).

Egy egész szám vagy 0, vagy egy nem nullával kezd?d?, tízes számrendszerbeli egész szám, vagy egy - jel, majd egy nem nullával kezd?d?, tízes számrendszerbeli egész szám.

A program m?ködésének lépései a következ?k:

1. Az aktuális kép el?készítése CANVAS 200 100 végrehajtásával.
2. Az alábbiak ismétlése, amíg van mit beolvasni a bemenetr?l:
    1. A bemenetr?l egy sort be kell olvasni, és azt utasításnak kell értelmezni.
    2. Ha a beolvasás során hiba keletkezik (pl. ismeretlen utasításnév, hiányzó paraméter, túl sok paraméter, PATH esetén páratlan sok vagy 6-nál kevesebb koordináta, hibás formátumú paraméter, fölösleges 0 az egész szám elején, ékezetes bet? egy névben, CANVAS esetén 1-nél kisebb szám), akkor SYNTAX hibát kell jelezni.
    3. Ha a PATH p, az RGB c vagy a SHAPE s neve már definiálva van, akkor DEFINED hibát kell jelezni. Egy név definiáltnak számít, ha egy korábbi tetsz?leges PATH, RGB vagy SHAPE parancs els? paramétere volt.
    4. Ha a SHAPE p vagy c neve, vagy a DRAW s neve még nincs definiálva, akkor UNDEFINED hibát kell jelezni.
    5. Ha a PATH-nak van olyan éle, amelynek vízszintessel bezárt szöge nem 45 fok többszöröse, akkor ANGLE hibát kell jelezni.
    6. Ha a PATH csúcspontjai nem konvex sokszöget definiálnak, akkor NOT_CONVEX hibát kell jelezni.
    7. Ha a SHAPE eltolt alakzata kilóg a képb?l, akkor BOUNDS hibát kell jelezni.
    8. Az utasítást a fent definiált módon végre kell hajtani.
3. Az aktuális képet a fent definiált formátumban ki kell írni a kimenetre.

A hibajelzés az alábbi módon történik. A program számolja, hány sort olvasott már be. Ha a futás során e hibát kell jelezni, amely az n-edik sor beolvasása után derült ki, akkor a program az e IN n hibaüzenetet írja ki egy soremeléssel lezárva, majd kilép (Ruby-ban Process.exit()). Példa hibaüzenet: NOT_CONVEX IN 42

Els? részfeladat: Írjon programot Ruby nyelven, amely a Draw45 rajzolási modellt valósítja meg, azzal az egyszer?sítéssel, hogy az utasításokat csak a SYNTAX ellen?rzésig bezárólag hatja végre. Tehát a program kimenete vagy egy 200x100-as fehér kép, vagy egy SYNTAX hibaüzenet.

Második részfeladat: Írjon programot Ruby nyelven, amely a Draw45 rajzolási modellt valósítja meg azzal az egyszer?sítéssel, hogy a teli kitöltöttség? alakzatokat nem rajzolja ki, tehát a SHAPE utasítás nem rajzol semmit, ha az f paramétere 1.

Harmadik részfeladat: Írjon programot Ruby nyelven, amely A Draw45 rajzolási modellt teljes mértékben megvalósítja.

Mintabemenet (a mintakimenet egy svájci zászló: piros alapon fehér kereszt)

```
RGB piros 218 22 3
RGB feher 254 255 252
CANVAS 125 125
PATH pirosnegyzet 0 0 124 0 124 124 0 124
PATH fekvoteglalap 21 50 103 50 103 74 21 74
```

```
PATH alloteglalap 50 21 74 21 74 103 50 103
SHAPE pirosnegyzets pirosnegyzet piros 1
SHAPE fekvoteglalaps fekvoteglalap feher 1
SHAPE fekvoteglalaps fekvoteglalap feher 1
DRAW pirosnegyzets 0 0
DRAW fekvoteglalaps 0 0
DRAW fekvoteglalaps 0 0
```

Verseny

Versenyfeladatok

A feladatok kiírásai és minták megtalálhatóak a SIO-ban:

- simpson
- atvalt
- test
- preps
- nemset

Megoldások

- simpson (Mikulán Attila megoldása):

```
#include<math.h>

double simpson(double f(double), double a, double b) {
  double k;
  k=(f(a)+4.0*f((a+b)/2)+f(b))*((b-a)/6);
  return(k);
}
```

- atvalt (Gubek Andrea megoldása)

```
#include <stdio.h>

int main() {
  int A, B, i=0;
  int n;
  while(2==scanf("http://wiki.math.bme.hu%d %d"http://wiki.math.bme.hu, &A, &B)) {
    int szam[256];
    n=0;
    for(i=0;i<256;i++) {szam[i]=0;}
    while (A>=B) {
      szam[n]=A%B;
      n++;
      A=A/B;
    }
    szam[n]=A;
    for(i=n;i>=0;i--) {
     printf("http://wiki.math.bme.hu%d"http://wiki.math.bme.hu, szam[i]);
    }
    printf("http://wiki.math.bme.hu\n"http://wiki.math.bme.hu);
  }
return 0;
}
```

- test (T?z Csaba megoldása):

```
class Test
  def felulet
    fail
  end
  def terfogat
    fail
  end
  def sulypont
    fail
  end
end

class Gomb< Test
  attr_accessor :r,:cx,:cy,:cz
  def felulet
    4*@r**2*Math::PI
  end
  def terfogat
    @r**3*Math::PI*4/3.0
  end
  def sulypont
    [@cx,@cy,@cz]
  end
end

class Teglatest< Test
  attr_accessor :ax,:ay,:az,:bx,:by,:bz
  def felulet
    a=(ax-bx).abs
    b=(ay-by).abs
    c=(az-bz).abs
    2*(a*b+a*c+b*c)
  end
  def terfogat
    a=(ax-bx).abs
    b=(ay-by).abs
    c=(az-bz).abs
    a*b*c
  end
  def sulypont
    [(ax+bx)/2.0,(ay+by)/2.0,(az+bz)/2.0]
  end
end
```

- preps
  - Szabó Viktor megoldása

```
b=0
forma=[]
formavan=0
uj=0
elso=1
STDIN.each_line {|s|
  if (s.split("http://wiki.math.bme.hu"http://wiki.math.bme.hu)[0]=="http://wiki.math.bme.hu#"http
  t=s.chomp.split
  if (t[0]=="http://wiki.math.bme.huPICTURE"http://wiki.math.bme.hu);
      print "http://wiki.math.bme.hu%!PS-Adobe-3.0 EPSF-2.0\n%%BoundingBox: #{t[1].to_i} #{t[2].to
    forma=[]
    formavan=0
    uj=1
  end
  if (t[0]=="http://wiki.math.bme.huDRAW"http://wiki.math.bme.hu);
    print "http://wiki.math.bme.hunewpath\n"http://wiki.math.bme.hu
```

```
    if (t[1]=="http://wiki.math.bme.huclosed"http://wiki.math.bme.hu or t[1]=="http://wiki.math.bm
      forma=t[1]
      formavan=1
      t.delete_at(1)
    end
    if (t[1].split("http://wiki.math.bme.hu"http://wiki.math.bme.hu)[0]=="http://wiki.math.bme.hu(
      szin1=t[1].split("http://wiki.math.bme.hu("http://wiki.math.bme.hu)[1]
      szin2=t[2]
      szin3=t[3].split("http://wiki.math.bme.hu)"http://wiki.math.bme.hu)[0]
      print "http://wiki.math.bme.hu#{szin1} #{szin2} #{szin3} setrgbcolor\n"http://wiki.math.bme.
      t.delete_at(1)
      t.delete_at(1)
      t.delete_at(1)
    end
    t.delete_at(0)
    print "http://wiki.math.bme.hu#{t[0]} #{t[1]} moveto\n"http://wiki.math.bme.hu
    b=2
    while (b<t.size);
      print "http://wiki.math.bme.hu#{t[b]} #{t[b+1]} lineto\n"http://wiki.math.bme.hu
      b+=2
    end
    if (forma=="http://wiki.math.bme.huclosed"http://wiki.math.bme.hu); print "http://wiki.math.bm
    if (formavan==0 or forma=="http://wiki.math.bme.huclosed"http://wiki.math.bme.hu); print "http
    if (forma=="http://wiki.math.bme.hufilled"http://wiki.math.bme.hu); print "http://wiki.math.bm
  end
}
print "http://wiki.math.bme.hugrestore\nshowpage\n%%EOF\n"http://wiki.math.bme.hu
```

- preps
  - Sisak Áron megoldása

```
STDIN.each_line {|l|
  begin
    # PICTURE
    if(/^PICTURE/.match(l))
      print "http://wiki.math.bme.hu%!PS-Adobe-3.0 EPSF-2.0\n"http://wiki.math.bme.hu
      print "http://wiki.math.bme.hu%%BoundingBox: "http://wiki.math.bme.hu + l.sub(/PICTURE\s+/,
      print "http://wiki.math.bme.hu%%EndComments\n"http://wiki.math.bme.hu
      print "http://wiki.math.bme.hugsave\n"http://wiki.math.bme.hu
    # DRAW
    elsif(/^DRAW/.match(l))
      type = "http://wiki.math.bme.hupoly"http://wiki.math.bme.hu
      if(/^DRAW filled/.match(l))
      type = "http://wiki.math.bme.hufilled"http://wiki.math.bme.hu
        l = l.sub(/DRAW filled/, '')
      elsif(/^DRAW closed/.match(l))
        type = "http://wiki.math.bme.huclosed"http://wiki.math.bme.hu
        l = l.sub(/DRAW closed/, '')
      end
      l = l.sub(/DRAW/, '')
      print "http://wiki.math.bme.hunewpath\n"http://wiki.math.bme.hu
      if(not (colors = l.scan(/\((.*)\)/)).empty?)
        print colors[0][0] + "http://wiki.math.bme.hu setrgbcolor\n"http://wiki.math.bme.hu
        l = l.sub(/\(.*\)/, '')
      end
      vertices = l.split("http://wiki.math.bme.hu\s"http://wiki.math.bme.hu) #.map! {|s| s.to_i}
      print vertices.shift + "http://wiki.math.bme.hu "http://wiki.math.bme.hu + vertices.shift +
      vertices.each_index {|i|
        print vertices[i] + "http://wiki.math.bme.hu "http://wiki.math.bme.hu + vertices[i+1] + "h
      }
      print "http://wiki.math.bme.huclosepath\n"http://wiki.math.bme.hu if type == "http://wiki.ma
      print (type == "http://wiki.math.bme.hufilled"http://wiki.math.bme.hu) ? "http://wiki.math.b
    # COMMENT
```

```
    elsif(/#/.match(l))
    # FORMAT VIOLATION
    else
      fail
    end
  end
}
print "http://wiki.math.bme.hugrestore\n"http://wiki.math.bme.hu
print "http://wiki.math.bme.hushowpage\n"http://wiki.math.bme.hu
print "http://wiki.math.bme.hu%%EOF\n"http://wiki.math.bme.hu
```

- nemset
  - Wettl Ferenc megoldása

```
#include <ctype.h>
#include <stdio.h>

char cards[81][4];                // a kártyák tárolására használt tömb
int num;                          // num = a beolvasott kártyák száma

int isset(int a, int b, int c) {  // ellen?rizzük, hogy a 3 kártya set-e
  int i;
  for(i=0; i!=4; i++) {
    if((cards[a][i] == cards[b][i] && cards[b][i] == cards[c][i]) ||
       (cards[a][i] != cards[b][i] && cards[a][i] != cards[c][i] && cards[b][i] != cards[c][i]))
      continue;
    return 0;
  }
  return 1;
}

int main(void) {
  int c, i, j, k, m, n, x;
  int lehet;                      // lehet-e találni még olyan kártyahalmazt
                                  //   amit ellen?rizni kell?
  int van;                        // van (=1) ha van SET a kiválasztott
                                  //   kártyahalmazban (egyébként =0)
  int set[81];                    // a vizsgált részhalmaz kártyáinak sorszámai

  num = 0;
  i = 0;
  while((c=getchar()) != EOF) {   // a kártyák beolvasása
    if(!isspace(c)) {
      cards[i/4][i%4] = c;
      i++;
    }
  }
  num = i / 4;
  lehet=1;
  m = 0;
  for (i=0;i<num;i++) {set[i]=0;}

  set[0]=0; set[1]=1; set[2]=2;   // válasszuk ki az els? 3 kártyát
  if (num>2) {n=3;} else {n=num;} // ide elég lenne n=3 is, de így 1 vagy 2
                                  //   kártyával is megy

  while (lehet) {                 // amíg lehet hozzávenni vagy
                                  //   kicserélni nagyobb sorszámúra
    van=0;
    for (i=0; i<n-2 && !van; i++) {
      for (j=i+1; j<n-1 && !van; j++) {
        if (isset(set[i],set[j],set[n-1])) {van=1; break;}
      }
```

```
    }
    if (!van && n>m) {            // m (az eddig talált legnagyobb
      m=n;                        //   SET-nélküli halmaz mérete)
    }

    if (set[n-1]<num-1) {         // a legnagyobb sorszámú nincs kiválasztva?
      if (van) {                  // van benne SET?
        set[n-1]+=1;              //   akkor az utolsó kártyát cseréljük ki
      }                           //   a következ?re.
      else {                      // nincs benne SET?
        n++;                      //   akkor vegyünk hozzá még egy kártyát
        set[n-1]=set[n-2]+1;
      }
    }
    else {                        // az utolsó kártya is ki van választva
      n--;                        //   akkor kidobjuk
      if (n>2) {                  //   és ha n>2
        set[n-1]+=1;              //     akkor vesszük a következ?t
      } else if (set[1]<num-2) {  //   egyébként ha n=2 és az 2. nem az utsó
        set[1]+=1;                //     vesszük a következ? kett?t
        set[2]=set[1]+1;
        n=3;
      }        else if (set[0]<num-3) {  // ha n=2 és az 1. nem az utolsó el?tti
        set[0]+=1;                //   vesszük a következ? hármat
        set[1]=set[0]+1;
        set[2]=set[0]+2;
        n=3;
      } else {                    // ha ide jutottunk, vége
        lehet=0;
      }
    }
  }
  printf("http://wiki.math.bme.hu%d\n"http://wiki.math.bme.hu,m);
  return 0;
}
```

- nemset
  - Szabó Péter "http://wiki.math.bme.hupts"http://wiki.math.bme.hu megoldása:

```
#include <stdio.h>
#include <string.h>

char kartyak[1000][6];
int ks;

char h[6];

void harmadik(char *a, char *b) {
  h[4]='\n';
  h[5]='\0';
  h[0] = (a[0]==b[0]) ? a[0]
       : (a[0]!='1' && b[0]!='1') ? '1'
       : (a[0]!='2' && b[0]!='2') ? '2' : '3';
  h[1] = (a[1]==b[1]) ? a[1]
       : (a[1]!='P' && b[1]!='P') ? 'P'
       : (a[1]!='Z' && b[1]!='Z') ? 'Z' : 'L';
  h[2] = (a[2]==b[2]) ? a[2]
       : (a[2]!='U' && b[2]!='U') ? 'U'
       : (a[2]!='F' && b[2]!='F') ? 'F' : 'K';
  h[3] = (a[3]==b[3]) ? a[3]
       : (a[3]!='R' && b[3]!='R') ? 'R'
       : (a[3]!='T' && b[3]!='T') ? 'T' : 'H';
}
```

```
void megold(void) {
  int i, j, k, m, s, maxs, limit;
  char *a, *b, ok;
  if (ks==0) return;
  if (ks<3) { printf("http://wiki.math.bme.hu%d\n"http://wiki.math.bme.hu, ks); return; }
  limit = (1<<ks);
  maxs=2;
  for (i=0; i<limit; i++) {
    ok=1;
    for (j=0; ok && j<ks; j++) {
      if (0==(i&(1<<j))) continue;
      a=kartyak[j];
      for (k=j+1; ok && k<ks; k++) {
        if (0==(i&(1<<k))) continue;
        b=kartyak[k];
        if (0==memcmp(a, b, 4)) continue;
        harmadik(a, b);
        /* printf("http://wiki.math.bme.hua=(%s) b=(%s) h=(%s)\n"http://wiki.math.bme.hu, a, b, h)
        for (m=0; m<ks; m++) {
          if (0==(i&(1<<m))) continue;
          if (0==memcmp(h, kartyak[m], 4)) { ok=0; break; }
        }
      }
    }
    if (ok) {
      s=0;
      for (j=0; j<ks; j++) {
        if (0!=(i&(1<<j))) ++s;
      }
      if (s>maxs) maxs=s;
    }
  }
  printf("http://wiki.math.bme.hu%d\n"http://wiki.math.bme.hu, maxs);
}

int main(int argc, char **argv) {
  (void)argc;
  (void)argv;
  ks=0;
  while (NULL!=fgets(kartyak[ks],6,stdin)) {
    if (kartyak[ks][0]=='\0' || kartyak[ks][0]=='\n') {
      megold();
      ks=0;
    } else {
      ks++;
    }
  }
  megold();
  return 0;
}
```