Megoldás. Közismert – de egy kis számolással könnyen ellenőrizhető is – hogy minden 3×3-as bűvös négyzet a következő alakú:

$x-y$	$x+y+z$	$x-z$
$x+y-z$	x	$x-y+z$
$x+z$	$x-y-z$	$x+y$

A feladat tehát egy háromváltozós lineáris egyenletrendszer megoldása az x, y, z változókra. Minden olyan cella, amibe számot írtak, egy egyenletet jelent.

A példaprogramban a jól ismert eliminációs módszert alkalmaztuk. A szokásos eliminációs módszerek háromféle lépésből állnak:

- egy egyenletet szorzunk/osztunk egy számmal úgy, hogy az egyik kiszemelt együttható értéke pontosan 1 legyen;
- egy egyenlethez hozzáadjuk egy másik egyenlet számszorosát úgy, hogy az egyik kiszemelt együtthatója 0 legyen;
- kicserélünk két egyenletet.

Ezekkel a lépésekkel az egyenletrendszert a következő alakra szeretnénk hozni:

$$\begin{aligned} 1x + 0y + 0z &= a, \\ 0x + 1y + 0z &= b, \\ 0x + 0y + 1z &= c. \end{aligned}$$

Mivel egyenletrendszerünk hiányos is lehet, a következő lépést is megengedjük:

- ha a megoldás nem lehet egyértelmű és ez nem okoz ellentmondást, akkor a rendszerhez hozzávehetjük az $x = 0$, $y = 0$ és $z = 0$ egyenletek valamelyikét.

Első lépésként az első egyenletet normáljuk, azaz végigosztjuk úgy, hogy az x együtthatója 1 legyen. Ha az x együtthatója 0, akkor az első egyenletet kicseréljük egy másikkal. Ha az x együtthatója mindenhol 0, akkor a rendszer nem lehet egyértelmű, és hozzávesszük a rendszerhez az $x = 0$ egyenletet.

Ezután a normált első egyenlet konstansszorosait vonjuk ki a többi egyenletből úgy, hogy az x együtthatója mindenhol kiessen.

A második lépésben az y együtthatóját normáljuk 1-re a második egyenletben, és a második egyenlet felhasználásával eliminájuk az y együtthatóját az összes többiből. Ha az y együtthatója a második egyenlettől kezdődően mindenhol 0, akkor a rendszerhez hozzávesszük az $y = 0$ egyenletet.

Végül ugyanez történik a z együtthatójával a harmadik egyenletben.

Ha háromnál több egyenletünk van, akkor a többi egyenlet az elimináció után csak $0x + 0y + 0z = d$ alakú lehet. Ha valamelyik d nem 0, akkor az egyenletrendszer ellentmondó volt.

A fenti algoritmust a következő Pascal programban valósítottuk meg.

```
{***********************************
* S. 10.                           *
* Mekk Elek 11. o. t.              *
* Nekeresd, Ezermesterképző Intézet *
* kaposzta@nekeresd.hu             *
***********************************}
{ Ezt a programot Linux alatt, FreePascal 1.0.10 fordítóval teszteltük. }

{A bűvös négyzetet a következő alakban keressük:
+----+----+----+
|  x-y  | x+y+z |  x-z  |
+----+----+----+
| x+y-z |   x   | x-y+z |
+----+----+----+
|  x+z  | x-y-z |  x+y  |
+----+----+----+

Ezeket az együtthatókat itt definiáljuk.
}
const
parameter_egyutthatok : array [1..3,1..3,1..3] of Real =
(( ( 1, -1,  0 ),  ( 1,  1,  1 ), ( 1,  0, -1 ) ),
```

```
( ( 1,  1, -1 ),  ( 1,  0,  0 ), ( 1, -1,  1 ) ),
( ( 1,  0,  1 ),  ( 1, -1, -1 ), ( 1,  1,  0 ) )   );

{ Egyenletek: Minden egyenlet Ax+By+Cz=D alakú. A négy együtthatót egy 4-elemű
vektorban tároljuk. Az egyenletek száma en, ami kezdetben legfeljebb 9. Mivel
közben is létrehozhatunk max. 3 egyenletet, 12-nek foglalunk helyet.  }
var
egyenlet : array[1..12,1..4] of Real;
en       : Integer; { az egyenletek tényleges száma }

{ A megoldás. }
var
mo          : array[1..3] of Real; { A keresett paraméterek }
mo_egyertelmu   : Boolean;             { Egyértelmű-e }
mo_ellentmondas : Boolean;             { Ellentmondó-e }

{ Az i-edik és a j-edik egyenlet felcserélése }
procedure EgyenletCsere( i : Integer; j: integer );
var
k : Integer;
x : Real;
begin
for k:=1 to 4 do
begin
x             := egyenlet[i,k];
egyenlet[i,k] := egyenlet[j,k];
egyenlet[j,k] := x;
end;
end; { EgyenletCsere }

{ Az input beolvasása }
procedure Beolvasas;
var
i,j,k,p,q : Integer;
s,t     : String;
begin
for i:=1 to 3 do
begin
ReadLn( s ); s:=s+' ';
q := 1; { A következő, még el nem olvasott karakter indexe }
for j:=1 to 3 do
begin
{ Megkeressük a következő karaktert, ami nem szóköz }
p := q; while (p<=Length(s)) and (s[p]=' ') do p:=p+1;
{ Megkeressük a következő szóközt avagy sor végét }
q := p; while (q<=Length(s)) and (s[q]<>' ') do q:=q+1;
t := Copy( s, p, q-p );
if (t<>'X') and (t<>'x') then
begin
{ Hozzáadunk a rendszerhez egy új egyenletet }
en := en+1;
egyenlet[en,1] := parameter_egyutthatok[i,j,1];
egyenlet[en,2] := parameter_egyutthatok[i,j,2];
egyenlet[en,3] := parameter_egyutthatok[i,j,3];
Val( t, egyenlet[en,4] );
end;
end;
end;
end; { Beolvasas }

{ Az egyenletrendszer megoldása }
```

```
procedure Szamolas;
var
i,j,v : Integer;
h       : Real;
begin
mo_egyertelmu   := true;
mo_ellentmondas := false;

{ Sorban eliminálunk mindegyik változó szerint. A v tartalmazza a
soron következő változó indexét. }
for v:=1 to 3 do
begin
{ Megkeressük az első olyan egyenletet, amiben a v-edik
együttható nem 0, és az egyenletet kicseréljük az n-edikkel }
i:=v; while (i<=en) and (abs(egyenlet[i,v])<1e-6) do i:=i+1;
if i>en then
begin
{ Ha nincs ilyen egyenlet, akkor csinálunk, a paraméter értéke 0. }
en := en+1;
egyenlet[en][1] := 0;
egyenlet[en][2] := 0;
egyenlet[en][3] := 0;
egyenlet[en][4] := 0;
egyenlet[en][v] := 1;
mo_egyertelmu   := false;
end;
{ Kicseréljük a két egyenletet }
if i<>v then EgyenletCsere( v, i );

{ Normáljuk az egyenletet }
h := egyenlet[v,v];
for j:=1 to 4 do egyenlet[v,j] := egyenlet[v,j]/h;

{ Eliminájuk a v-edik együtthatót az összes többi egyenletből }
for i:=1 to en do if i<>v then
begin
{ Az i-edik egyenletből kivonjuk a v-edik h-szorosát
úgy, hogy a v-edik együttható kiessen }
h := egyenlet[i,v];
for j:=1 to 4 do egyenlet[i,j] := egyenlet[i,j] - h*egyenlet[v,j];
end;
end;

{ Ellenőrizzük a további, teljesen eliminált egyenleteket, hogy
kiestek-e a konstansok }
for i:=4 to en do
if abs(egyenlet[i,4])>1e-6 then
begin
{ A konstans nem esett ki, az egyenlet 0x+0y+0z=c alakú }
mo_ellentmondas := true;
Exit;
end;

{ Kiolvassuk a megoldásokat }
for v:=1 to 3 do mo[v] := egyenlet[v][4];
end; { Szamolas }

{ Az eredmény kiíratása }
procedure Kiiratas;
var
i,j : Integer;
```

```
c   :  Real;
begin
if mo_ellentmondas then
WriteLn( 'Nincs megoldás' )
else
begin
for i:=1 to 3 do
begin
for j:=1 to 3 do
begin
{ Kiírjuk az i-edik sor j-edik elemét. }
{ A mező értéke a megadott módon felírva a paraméterekkel: }
c := parameter_egyutthatok[i,j,1] * mo[1] +
parameter_egyutthatok[i,j,2] * mo[2] +
parameter_egyutthatok[i,j,3] * mo[3];
if abs(c)<1e-6 then c:=0;
if frac(c+1e-6)<2e-6 then
Write( c:0:0 )  { egész, nem kellenek tizedesjegyek }
else
Write( c:0:2 ); { nem egész, 2 tizedesjegyre kerekítjük }
if j<3 then Write( ' ' ) else WriteLn;
end;
end;

if not mo_egyertelmu then WriteLn( 'A megoldás nem egyértelmű' );
end;
end; { Kiiratas }

begin
Beolvasas;
Szamolas;
Kiiratas;
end.
```

A beérkezett dolgozatok tanulságai

A programokat összesen 1023 tesztadaton próbáltuk ki. A 9 mező összes részhalmazát kétféleképpen is kitöltöttük (kivéve az üres halmazt, amit csak egyféleképpen lehet). Ahol lehetett, a kétféle kitöltés közül az egyik megoldható volt, a másik ellentmondásos.

Sajnos több program sem az előírt formában várta az adatokat és az eredményt nem a kért formában írta ki, ami az automatizált tesztelést megnehezítette. Kérjük, figyeljetek oda jobban, hogy a feladat szövege milyen formátumokat ír elő. Ez nem akadékoskodás, így tudjuk elvégezni a munkánkat.

Nagyon sok programhoz nem érkezett semmilyen leírás, esetenként még megjegyzések sem voltak a kódban. Pedig a maximális pontszám eléréséhez szükséges, hogy a megvalósított algoritmust röviden írjátok le, és a programkódban elhelyezett kommentekből kiderüljön, hogy az egyes eljárásoknak és a fontosabb változóknak mi a tartalma. A dokumentáció nélküli megoldások legfeljebb 7 pontot kaphattak.

Gyakori probléma volt a struktúrálatlanság. Az eljárások nem különültek el (pl. nem volt közöttük üres sor sem), vagy éppen sokszorosan egymásba skatulyázott blokkok szerepeltek a kódban. A programokat érdemes kisebb eljárásokra bontani; akkor olvashatóbbak, az egyes eljárásokat külön-külön tudjátok tesztelni és a hibákat is sokkal könnyebb megtalálni.