I. 469.

Horcsin Bálint
Technikai információ:
Programnyelv: Java 1.8
Futtatható main(String[] args) függvény az I469 osztályban található.
Fejlesztői környezet: Eclipse Photon Release (4.8.0)

Dokumentációm elején szeretném jelezni, hogy a feladatra minden esetben van véleményem szerint megoldás. Egy lehetséges megoldási menet (melyet a programom is használ) az alább olvasható.

A feladat megoldásánál nem volt célom a legkevesebb lépésszám megkeresése, igazából a legegyszerűbb és legbonyolultabb esetekre is kb. 100-400-at lép.
Egy előre meghatározott módon rakja össze a megoldást. - Nem rekurzív, hogy a futásidőt csökkentsem.
Ez olyan, mint egy rubik kocka. Többféleképpen meg lehet oldani, és az átlagos emberek nagy lépésszámú megoldásokat készítenek.
A feladat nem kötötte ki, hogy milyennek kell lennie, így ez is megfelel a feltételeknek.

Első lépésben a táblázat ezen (sárga) részét oldja meg:

	F	G	H
	J	K	L

Minden egyes oszlophoz először elhelyezi a bal felső, majd felső sor 2. elemébe a beillesztendő számokat, aztán azokat belerakja a megfelelő oszlopba (Functions osztály segítségével).

x	y		
		x	
		y	

Ezt előre elkészített lépések végzik el. (További információ a forráskódban.)
Ezek után a felső sort rendezem el DCBA sorrendben: Mindig kiszedem belőle a következőt az első sort használva, aztán elhelyezem a megfelelő helyre.

A	B	C	D
	F	G	H
	J	K	L

Ezt előre elkészített lépések végzik el. (További információ a forráskódban.) Ezek után az első oszlop lehet, hogy jó lesz, lehet, hogy nem. Jó:

A	B	C	D
E	F	G	H
I	J	K	L

Rossz:

A	B	C	D
I	F	G	H

I. 469.

Horcsin Bálint
Budapest, Németh László Gimn., 10. o. t.
horcsinbalint@gmail.com

E	J	K	L

Ha rossz, akkor a „s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 o 1 o 1 o 1 o 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 o 1 o 1 s 1 s 1 s 1 o 1 o 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 o 1 o 1 o 1 o 1 s 1 s 1 s 1 o 1 o 1 o 1 o 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 o 2 o 2 s 1 s 1 s 1 o 2 o 2 o 2 o 2 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 o 2 o 2 o 2 o 2 o 2 o 2 o 3 o 3 s 1 s 1 s 1 o 3 o 3 s 1 s 1 s 1 o 2 o 2 s 1 s 1 s 1 s 1 s 1 o 3 o 3 o 4 o 4 s 1 s 1 s 1 o 4 o 4 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 s 1 o 3 o 3 s 1 s 1 s 1 s 1 s 1 s 1 o 4 o 4 o 4 o 4 o 4 o 4 ” lépéssorozattal kihozható a mintabemenet.[1] Abból pedig megoldható a feladat az „o 4 s 3 o 2 s 1” lépéssorozattal. (Most nem tördeltem, a program a kimenetét tördelni fogja.) Megj.: Ezt a program már csak kiírja, nem szimulálja le.

Mivel ez egy általános módszer, így minden esetben van megoldása a programnak, soha nincs ~~1~~es eset.

Fejlesztői dokumentáció:
Coord osztály: egy pont koordinátáit lehet vele továbbadni.
Functions osztály:

- •: `public static void felsoelso(Helyzet h, int x, int y)` az adott x-y koordinátán lévő mezőt a bal felső sarokba teszi. (felviszi a mezőt a felső sorba, utána a felső sort mozgatja.)
- •: `public static void felsomasodik(Helyzet h, int x, int y)` az adott x-y koordinátán lévő mezőt a felső 2. mezőre teszi. Ha a felső sorban van, lejjebb tolja, majd mozgatja a felső sort, hogy a felsoelso után kerüljön mindenképp, és utána felrakja a felső sorba. Majd pedig a felső sort eltolja, hogy jó helyen kezdődjön.

•: a `public static void push(Helyzet h, int oszlop)` felsoelsot, és felsomasodikat a megadott oszlopba tolja Helyzet osztály:

Az aktuális helyzetet tartalmazza. A függvényeit nem dokumentálom, mivel a feladat ismeretében értelmezhetőek.
I469 osztály:

A megoldás vezérlését végzi.

Test osztály:

Egy kódba írt helyzeten a standard bemeneten megadott lépéseket végzi el. - Debug-oláshoz használt kód.

[1] A programot lefuttattam a mintabemenetre, kicserélgettem az utasításokat, hogy visszafelé hozza ki.