Tomi az interneten talált leírás alapján egy számlálót épített. A számláló – a leírás szerint – 0 másodperctől 999 másodpercig számlált volna, másodpercenként eggyel növelve a kiírt értéket. Azonban valamit elronthatott, mert az eszköz tizedmásodpercenként váltott. A hétszegmenses folyadékkristályos kijelző ráadásul nem mindig reagált a megfelelő gyorsasággal a vezérlésre, előfordult, hogy azonnal váltott egy szegmens, míg máskor várni kellett, de legfeljebb k tizedmásodpercet. Ha a k értéke 2, és egy szegmens be kell kapcsoljon, amely korábban nem világított, lehet, hogy a 0., lehet, hogy az 1., de az is lehet, hogy a 2. tizedmásodpercben kapcsol majd be. A bekapcsolás nem szükségszerűen történik meg, ha a következő jel éppen a kikapcsolás és már az jut érvényre.

Tomi elgondolkodott azon, hogy ha nem ismerjük a kezdő állapotot, és a számlálót sem állíthatjuk meg, ki lehet-e találni, hogy éppen melyik értéket kellene mutatnia az utolsó számjegynek. Rövidesen arra a megállapításra jutott, hogy bár a hiba eltérő késleltetéssel jelentkezik, megfelelő számú állapot vizsgálatával kikövetkeztethető az aktuális állás.

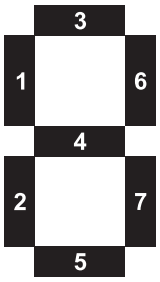
Írjunk olyan programot, amely az egymást követő állapotok ismeretében képes az aktuális számjegy meghatározására!

A bemeneti fájl első sorában egy egész szám, a kijelző reakcióideje szerepeljen ($0 \leq k \leq 5$). A következő sorban a megfigyelések n száma legyen olvasható ($1 \leq n \leq 100$). Az ezt követő n sor a kijelző utolsó jegyének állapotát a megfigyelés kezdetétől tizedmásodpercenként adja meg. Egy-egy sor az adott pillanatban írja le a kijelző utolsó jegyét, megadva az egyes szegmensek állapotát (0 – nem világít, 1 – világít). A kimeneti fájlban meg kell adni, hogy hány állapot vizsgálata után állapítható meg a kijelző állása és akkor éppen milyen értéket kellene mutatnia. Ha a megfigyelés időtartama nem lenne elegendő a válasz megadására, a kimenetnek a 0 értéket kell tartalmaznia.

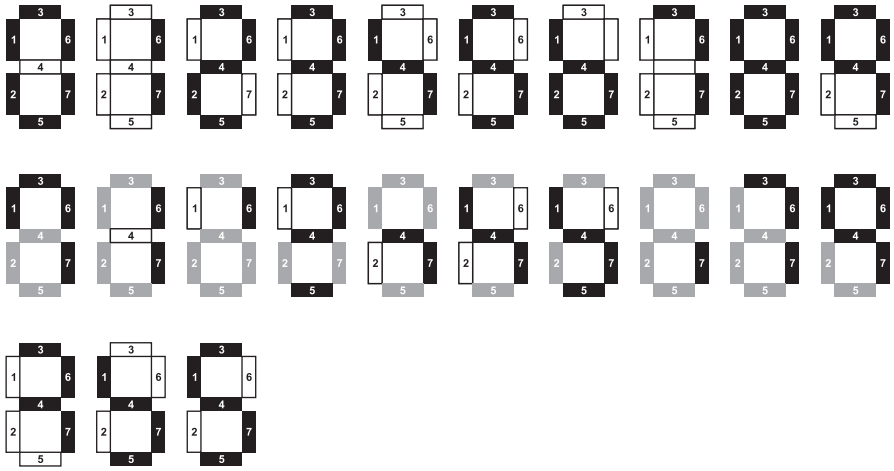
A számjegyeket felépítő szegmensek állapotainak megadása az alábbi *ábra* szerinti sorrendben történik:

Az egyes számjegyek kódolása:

```
0: 1 1 1 0 1 1 1
1: 0 0 0 0 0 1 1
2: 0 1 1 1 1 1 0
3: 0 0 1 1 1 1 1
4: 1 0 0 1 0 0 1
5: 1 0 1 1 1 0 1
6: 1 1 0 1 1 0 1
7: 0 0 1 0 0 1 1
8: 1 1 1 1 1 1 1
9: 1 0 1 1 0 1 1
```



Bemenet	Kimenet
1 6 0 0 1 1 0 1 1 1 0 0 1 1 0 1 1 0 1 1 1 0 1 0 0 1 1 0 0 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1	3 6



A fenti *ábra* első sora a késleltetés nélküli kijelzőállapotokat mutatja 0-tól 9-ig. A második sorban a példa a $k = 1$ késleltetés szerint mutatja a kijelzőt. A fekete szegmensek az adott számjegy esetén mindenképpen világítanak, a fehérek pedig semmiképpen sem. A szürke színű szegmensek állapota nem határozható meg. A harmadik sor fekete szegmensei a példa első három állapotát mutatják, amelyből már egyértelműen meghatározható az utolsó számjegy valós értéke.

A program első parancssori argumentuma a bemeneti fájl neve, a második argumentuma pedig a kimeneti fájl neve legyen.

Beküldendő a program forráskódja (`i304.pas`, `i304.cpp`, ...), valamint a program rövid dokumentációja (`i304.txt`, `i304.pdf`, ...), amely tartalmazza a megoldás rövid leírását, és megadja, hogy a forrásállomány melyik fejlesztő környezetben fordítható.