A *Hamis dobókocka feladat* két korábbi középszintű digitális kultúra érettségi feladaton alapszik: a 2023. májusi feladatsor táblázatkezelés témájú *Pénzfeldobás*, illetve a 2022. októberi feladatsor programozás témájú *Kockák* feladatán.

Hamis dobókocka

Egy játékgyártó üzem dobókockákat készít társasjátékokhoz. Mielőtt forgalomba hozzák a kockákat, több ezer dobással tesztelik azokat. Egyik kockájukat inhomogén tömegeloszlása miatt selejtezniük kell, a tesztelés alapján ugyanis 4-est 23% eséllyel, 5-öst 19%, 1-est 16%, a további három számot pedig egyaránt 14% eséllyel lehet dobni.

Modellezzük ennek a hamis kockának a tesztjét véletlenszám generátorral!

1. Végezzünk 1000 dobásos tesztet úgy, hogy az egyes számok elméleti előfordulási esélye egyezzen a megadott értékekkel!

2. Adjuk meg, hogy egy 1000-es dobás sorozaton belül ténylegesen mekkora százalékban fordultak elő az egyes számok!

3. Adjuk meg a leghosszabb homogén dobási sorozat adatait (azaz annak a sorozatnak az adatait, amikor legtöbbször dobtuk egymás után ugyanazt a számot):

- milyen hosszú a sorozat?

- melyik számmal dobtuk a sorozatot?

- hányadik dobástól hányadik dobásig tartott a sorozat?

A képernyőn az alábbi mintának megfelelően jelenjenek meg a kimeneti eredmények:

Számok előfordulása:

1 – 15,2% 2 – 14,3% 3 – 12,9% 4 – 25,8% 5 – 17,8% 6 – 14,0%

Leghosszabb homogén sorozat hossza: 8

Sorozatban szereplő szám: 4

Sorozat kezdete: 812. dobás

Sorozat vége: 819. dobás

Az alábbiakban megvizsgáljuk, hogyan lehet kialakítani azt az algoritmust, amely képes megadni a leghosszabb homogén dobás sorozat adatait.

Egy algoritmus létrehozásában sokszor jelenthet segítséget, ha sikerül elemi lépésekből álló cselekvési tervet készítenünk. Vegyünk tehát elő papírt és ceruzát, majd írjunk a papírra egy megfelelően rövid véletlen sorozatot. A rövid sorozat választásának praktikus oka van: szeretnénk könnyen áttekinteni a dobás sorozat elemeit. Legyenek a véletlenszerűen választott számok a következők: 2, 4, 4, 3, 1, 1, 6, 4, 5, 5, 5, 2.

Rápillantva a sorozat elemeire egy szemvillanás alatt ki tudjuk választani a leghosszabb homogén sorozatot. Nem szabad elfelejtenünk azonban, hogy a számítógép processzora nem egyben „látja” a teljes sorozatot, a sorozat elemeit csak a beolvasás sorrendjében, egyesével képes megvizsgálni! A processzor működését lekövető cselekvési modellben ezért a sorozat tagjait sorszámmal ellátott lezárt dobozokba képzeljük, minden egyes dobozban pontosan egy dobásértéket elhelyezve. A sorszámok alapján egymás után nyithatjuk ki a dobozokat, megismerve ezzel egy-egy doboz tartalmát. A cselekvési terv megalkotásában idáig jutva fontos elvi döntést kell meghoznunk: hány doboz legyen egyszerre nyitva?

Próbálkozzunk és nyissunk ki először egy dobozt, az elsőt! Ez a doboz az első kockadobás eredményét (d_1) tartalmazza! Megállapíthatjuk, hogy $d_1=2$. Mit tehetünk ezután? Rá kell jönnünk, ez az információ kevés, nem tudunk döntést hozni, ki kell tehát nyitni a következő dobozt is. A második doboz tartalma $d_2=4$. Most már meghozhatjuk a megfelelő döntést: **ha** $d_2=d_1$, **akkor** eljutottunk egy kettő hosszúságú homogén részsorozathoz, **egyébként** $d_2\neq d_1$, tehát az eddig talált leghosszabb homogén „sorozat” hossza továbbra is csak egy. Kinyithatjuk a következő dobozt.

Az első elemi lépés sikeres azonosítása után több logikai következménnyel is szembesülünk. Első következmény: mivel végig kell lépkedni egymás után a sorozat összes elemén, mindenképpen számlálós ciklusba kell szervezni a lépéseket, a ciklusszámlálót pedig a kettes számmal kell indítanunk (az első dobozt nyitottnak feltételezzük), és egyesével sorozat elemszámig kell léptetnünk (a példában 12-ig). Második következmény: ha automatizálni szeretnénk az összehasonlítás műveletét, akkor általános indexelést kell használnunk: d_i , d_{i-1}. Harmadik következmény: az első elemi lépés nyelvi megfogalmazásában tükröződik a lépés logikai szerkezete (ha – akkor – egyébként), tehát a számlálós ciklus belsejében elágazást kell alkalmaznunk. Foglaljuk össze eddigi eredményeinket:

```
Ciklus i=2-12
  Ha di=di-1 akkor
    hossz=2 (?)
  egyébként
    (?)
  Elágazás vége
Ciklus vége
```

Látható, hogy már az első elemi lépés azonosításával is jelentős előrehaladást sikerült elérnünk az algoritmus szerkezetének kialakításában. Persze vannak még bizonytalan részletek.

Egy elágazás akkor ágában előfordulhat konstans értékű művelet, azonban mindez egy ciklus belsejében van! Nem tűnik értelmes választásnak újra és újra konstans értékkel inicializálni a hossz változót. Ráadásul, hogy fog ekkor megjelenni a helyes *hossz=3* eredmény?

Javítsuk az algoritmust! Mivel egy sorozat hossza a feladat értelmezése alapján minden esetben legalább egy, ezért nyugodtan előre rögzíthetjük ezt az értéket. Az akkor ágban ezt követően a homogén sorozat hosszának eggyel történő növelése helyes eredményt fog adni az elágazási feltétel teljesülése esetén.

```
hossz=1
Ciklus i=2-12
  Ha di=di-1 akkor
    hossz=hossz+1
  egyébként
    (?)
  Elágazás vége
Ciklus vége
```

Eddig megalkotott algoritmusunk *i=4* esetén fordulóponthoz ér! Nem fordult még elő ugyanis olyan eset, hogy az egyébként ág tartalmának „lebegtetése” problémát okozott volna (pl. i=2 esetén). Most azonban ismét az egyébként ágba lépünk, viszont van már egy kettő hosszú homogén sorozatunk. Ha most nem állítanánk alaphelyzetbe a hossz változót (hossz=1), akkor i=6 esetén $d_6=d_5$ miatt a hossz változó értéke 3-ra nőne, pedig ilyen hosszú homogén sorozatunk még nem is volt.

```
hossz=1
Ciklus i=2-12
  Ha di=di-1 akkor
    hossz=hossz+1
  egyébként
    hossz=1
  Elágazás vége
Ciklus vége
```

Kezd kiteljesedni algoritmusunk, már nem kell „lebegtetnünk” az egyébként ág tartalmát. Látható, hogy a hossz változó értékének alaphelyzetbe állítása ellenére is meg fogjuk találni a három hosszú homogén sorozatot, ugyanis i=11 esetében hossz=3 teljesül. Van azonban egy utolsó lépés (i=12), és ekkor $d_{12}\neq d_{11}$ miatt sajnos újra elvesztjük a hossz változó 3-as értékét! Mielőtt újra alaphelyzetbe állítjuk a hossz változó értékét, ki kell tehát menteni a korábban felvett értéket!

```
hossz=1
Ciklus i=2-12
  Ha di=di-1 akkor
    hossz=hossz+1
  egyébként
    véglegeshossz=hossz
    hossz=1
  Elágazás vége
Ciklus vége
```

Módosítsuk példabeli sorozatunkat, és változtassuk meg a leghosszabb homogén sorozat helyzetét: 2, 4, 4, 3, 5, 5, 5, 6, 4, 1, 1, 2.

Ebben az esetben újra elvesztjük a helyes eredményt, hiszen *véglegeshossz* értéke először 2, majd 3, végül újra 2 lesz! A véglegeshossz változó értékét tehát csak akkor szabad felülírni, ha az aktuális hossz érték nagyobb a véglegeshossz értékénél! Jó, ha egy névváltoztatással utalunk is a véglegeshossz változó szerepére: legyen tehát a véglegeshossz változó neve *maxhossz*!

```
hossz=1, maxhossz=1
Ciklus i=2-12
  Ha di=di-1 akkor
    hossz=hossz+1
  egyébként
    Ha hossz>maxhossz akkor
      maxhossz=hossz
    Elágazás vége
    hossz=1
  Elágazás vége
Ciklus vége
```

Kiegészítés: szeretnénk hozzákapcsolni a maxhossz értékének kialakításához a szükséges további információkat is!

```
hossz=1, maxhossz=1
Ciklus i=2-12
  Ha di=di-1 akkor
    hossz=hossz+1
  egyébként
    Ha hossz>maxhossz akkor
      maxhossz=hossz
      szám=di-1
      végpozíció=i-1
    Elágazás vége
    hossz=1
  Elágazás vége
Ciklus vége
Ki: „Leghosszabb homogén sorozat hossza: ” maxhossz
    „Sorozatban szereplő szám: ” szám
    „Sorozat kezdete: ” végpozíció-maxhossz+1 „. dobás”
    „Sorozat vége: ” végpozíció „. dobás”
```

Úgy érezzük algoritmusunk tartalmilag teljessé vált, a homogén szekvencia helyzetétől függetlenül helyesen határozza meg a kimeneti értékeket, belefoghatunk a kódírásba. A kód elkészülte után megkezdjük a tesztelést, azonban néhány futtatás után a következő eredményt kapjuk:

```
3 5 6 1 4 6 1 5 2 3 2 2
Leghosszabb homogén sorozat hossza: 1
Sorozatban szereplő szám: 1
Sorozat kezdete: 1. dobás
Sorozat vége: 1. dobás !
```

Úgy tűnik, algoritmusunk nem érzékeli az utolsó pozíciókban megjelenő homogén szekvenciát, ezáltal a kimeneti változók is alapértelmezett értékükkel jelennek meg! Erre nem gondoltunk! Mi lehet a hiba?

Az algoritmus elemzése alapján észrevehetjük, hogy a homogén szekvencia létrejöttét a szekvenciából való kilépés pillanatában nyugtázzuk, pedig a szekvencia nem a kilépés pillanatában, a szekvencia tartalmától különböző elem megjelenésekor készül el, hanem valójában már akkor, amikor az utolsó azonos értékű elemet hozzáillesztjük a szekvenciához. Az igaz, hogy azt még nem tudjuk, folytatódni fog-e a szekvencia, de ettől az már akkor és ott létezik. Létrejöttét tehát akkor és ott kell nyugtázni, amikor és ahol létrejön!

```
hossz=1, maxhossz=1
Ciklus i=2-12
  Ha di=di-1 akkor
    hossz=hossz+1
    Ha hossz>maxhossz akkor
      maxhossz=hossz
      szám=di
      végpozíció=i
    Elágazás vége
  egyébként
    hossz=1
  Elágazás vége
Ciklus vége
```

```
4 2 4 5 2 1 1 4 2 4 4 4
Leghosszabb homogén sorozat hossza: 3
Sorozatban szereplő szám: 4
Sorozat kezdete: 10. dobás
Sorozat vége: 12. dobás
```