**Itt vannak a múlt heti házi feladatok, megoldásokkal együtt.**

1. **Mit ír ki? (6 pont)**

Legyen a következő program. Állapítsátok meg, mit ír ki a program a vég­rehajtás eredményeként.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 7;11 | 6;9 | 7;9 | | | 7;12 |
| **C** | | | | | | |
| **#include <stdio.h>**  **int** feldVektor(**int** v[], **int** \*n){  **int** s = 0;  **int** i = 2;  **while** (i <= \*n) {  s = s + v[i] - v[i - 1];  **if** (v[i] == v[i - 1])  \*n = \*n - 1;  i++;  }  **return** s;  } | | | **int** main(){  **int** v[8];  v[1] = 1; v[2] = 4;  v[3] = 2; v[4] = 3;  v[5] = 3; v[6] = 10;  v[7] = 12;  **int** n = 7;  **int** eredmeny = feldVektor(v, &n);  printf("%d;%d", n, eredmeny);  **return** 0;  } | | | |
| **C++** | | | | | | |
| **#include <iostream>**  **using namespace std;**  **int** feldVektor(**int** v[], **int** &n){  **int** s = 0;  **int** i = 2;  **while** (i <= n) {  s = s + v[i] - v[i - 1];  **if** (v[i] == v[i - 1])  n--;  i++;  }  **return** s;  } | | | **int** main(){  **int** v[8];  v[1] = 1; v[2] = 4;  v[3] = 2; v[4] = 3;  v[5] = 3; v[6] = 10;  v[7] = 12;  **int** n = 7;  **int** eredmeny = feldVektor(v, n);  cout << n << ";" << eredmeny;  **return** 0;  } | | | |
| **Pascal** | | | | | | |
| **type** vector = **array**[1..10] **of** integer;  **function** feldVektor(v: vector;  **var** n: integer): integer;  **var** s, i: integer;  **begin**  s := 0; i := 2;  **while** (i <= n) **do begin**  s := s + v[i] - v[i - 1];  **if** (v[i] = v[i - 1]) **then**  n := n - 1;  i := i + 1;  feldVektor := s;  **end**;  **end**; | | | | **var** n, eredmeny: integer; v: vector;  **Begin**  n := 7; v[1] := 1; v[2] := 4;  v[3] := 2; v[4] := 3; v[5] := 3;  v[6] := 10; v[7] := 12;  eredmeny := feldVektor(v, n);  write(n, ';', eredmeny);  **End**. | | |

**Megoldás**

Adott egy alprogram és az alprogramot meghívó programegység. A kérdés egyszerű, hiszen csak azt kell kiválasztanunk (a lehetséges változatok közül), hogy mit ír ki a hívó programegység. Ez a feladattípus valóban könnyű, de fi­gyelni, koncentrálni elengedhetetlen: észre kell vennünk a paraméterátadás mi­kéntjét (*érték szerint*, *cím szerint,* esetleg *referencia szerint*), és az értékek válto­zását is figyelmesen kell követnünk. Érdemes táblázatot készíteni. A versenyző a három programozási nyelvből azt választja ki, amelyiket a legjobban ismeri.

A program rögzíti a bemenet értékeit: ***n*** = 7, ***v*** = (1, 4, 2, 3, 3, 10, 12). Észre­vesszük, hogy az alprogram kiszámítja a tömb elemeinek összegét a másodiktól kezdve, de az összegből minden lépésben kivonja az aktuális elem előtti elem értékét. Tehát: (4 – 1) + (2 – 4) + (3 – 2) + (3 – 3) + (10 – 3) + (12 – 10) = 3 – 2 + 1 + 0 + 7 + 2 = 11. Ugyanakkor azt is fontos észrevennünk, hogy a sorozatban létezik két egymás után elhelyezkedő azonos értékű elem, tehát ***n*** értéke egyszer csökken. Így (mivel csökken a sorozat azon része, amit fel kell dolgoznunk) az utolsó tagot (2) nem adjuk az összeghez. A program kiírja ***n*** megváltozott értékét (6) és a kiszámolt eredményt (9). Tehát a helyes válasz: **B**.

1. **Logikai kifejezés (6 pont)**

Állapítsátok meg, hogy a következő kifejezések közül melyiknek lesz az érté­ke akkor és csakis akkor igaz, ha az ***n*** természetes szám osztható 3-mal és az utolsó számjegye 4 vagy 6:

n **DIV** 3 = 0 **és** (n **MOD** 10 = 4 **vagy** n **MOD** 10 = 6)

n **MOD** 3 = 0 **és** (n **MOD** 10 = 4 **vagy** n **MOD** 10 = 6)

(n **MOD** 3 = 0 **és** n **MOD** 10 = 4) **vagy** (n **MOD** 3 = 0 **és** n **MOD** 10 = 6)

(n **MOD** 3 = 0 **és** n **MOD** 10 = 4) **vagy** n **MOD** 10 = 6

**Megoldás**

Adott több logikai kifejezés, amelyekben a részkifejezések relációs, illetve aritmetikai kifejezések. A megoldáshoz szükséges ismerni a logikai és aritmetikai operátorokat, valamint az operátorok precedenciáját. Ahhoz, hogy a feltett kér­désre válaszolhassunk, kiértékeljük a kifejezéseket, hiszen több jó megoldás is előfordulhat.

Választunk – tetszőlegesen – egy természetes számot, amely osztható 3-mal és az utolsó számjegye 4 vagy 6. Elvégezzük az aritmetikai műveleteket, majd a kapott részeredményekre elvégezzük a logikai műveleteket. A zárójelekben talál­ható kifejezéseket kiértékeljük, és a kapott részeredményekkel elvégezzük a lo­gikai műveleteket.

Legyen, például ***n*** = 24 (utolsó számjegye 4) ⇒ 24 **DIV** 3 = 8, tehát nem 0 ⇒ a (24 **DIV** 3 = 0) kifejezés értéke *hamis*. Mivel a következő operátor az „**és**”, nincs értelme kiértékelni a következő részkifejezést (tudjuk, hogy (*hamis* **és** *hamis*) = *hamis*, valamint (*hamis* **és** *igaz*) = *hamis*). Az **A.** választ kizárjuk a jó válaszok közül. Ha ***n*** = 36 (utolsó számjegye 6), ugyanerre az eredményre jutunk.

A **B.** kifejezés esetében az első részkifejezést 24-re és 36-ra is igaznak találjuk, mivel 24 **MOD** 3 = 0 és 36 **MOD** 3 = 0. Ebben az esetben ki kell számolnunk a második részkifejezés értékét is: (24 **MOD** 10 = 4 **vagy** 24 **MOD** 10 = 6). Ha ***n*** = 24, az első részkifejezés *igaz*, és ha ***n*** = 36, *igaz* a második részkifejezés. Mivel az őket összekötő operátor a „**vagy**”, és tudjuk, hogy (*igaz* **vagy** *hamis* = *igaz*, illetve *hamis* **vagy** *igaz* = *igaz*), végrehajtjuk a zárójel előtti **és** műveletet: (*igaz* **és** *igaz* = *igaz*), így a **B.** választ jónak minősítjük.

Hasonlóképpen kiértékeljük a **C.** kifejezést: ha ***n*** = 24, az első zárójelben ta­lálható részkifejezés értéke *igaz*, a második *hamis*, de a részeredmény *igaz*. Mivel a következő művelet „**vagy**”, a második kifejezést nem értékeljük ki, és levonjuk a következtetést, hogy az eredmény *igaz*. Ha ***n*** = 36, a második részkifejezés értéke *hamis*, tehát az első zárójelben levő részkifejezés értéke *hamis*. Most fontos kiértékelni a második zárójelben levő részkifejezés értékét is. A részeredmény *igaz*, tehát a végeredmény is *igaz*. A **C.** választ is jónak találjuk. A megfelelő számolásokkal a **D.** választ *hamis*-nak találjuk, tehát a jó válaszok: **B.** és **C**.

1. **Csonkított számok összege (6 pont)**

Definiáljuk a számjegyekből álló ***k*** természetes számra a *csonkítás* műveletet: .

Állapítsátok meg, hogy az alábbi algoritmusok közül melyik határozza meg az ***n*** elemű ***x*** sorozat *csonkított elemeinek összegét*. Az elemek természetes szá­mok és kisebbek, mint 1 000 000 (***n*** – természetes szám és 1 ≤ ***n*** ≤ 1 000). Például, ha ***n*** = 4 és ***x*** = (213, 7, 78347, 22), akkor a csonkított elemek összege 21 + 0 + 78 + 22 = 121.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A.** | **Algoritmus** csonkítás(n, x)  s ← 0  **Amíg** n > 0 **végezd el**  **Ha** x[n] > 9 **akkor**  **Amíg** x[n] > 99 **végezd el**  x[n] ← x[n] **DIV** 10  **vége(amíg)**  s ← s + x[n]  **vége(ha)**  n ← n - 1  **vége(amíg)**  **térítsd** s  **Vége(algoritmus)** | **B.** | **Algoritmus** csonkítás(n, x)  s ← n  **Amíg** n > 0 **végezd el**  **Ha** x[n] > 9 **akkor**  **Amíg** x[n] > 99 **végezd el**  x[n] ← x[n] **DIV** 10  **vége(amíg)**  s ← s + x[n]  **vége(ha)**  n ← n - 1  **vége(amíg)**  **térítsd** s  **Vége(algoritmus)** | |
| **C.** | **Algoritmus** csonkítás(n, x)  s ← 0  **Amíg** n > 0 **végezd el**  **Ha** x[n] > 9 **akkor**  **Amíg** x[n] > 99 **végezd el**  x[n] ← x[n] **DIV** 10  s ← s + x[n]  **vége(amíg)**  **vége(ha)**  n ← n - 1  **vége(amíg)**  **térítsd** s  **Vége(algoritmus)** | **D.** | | **Algoritmus** csonkítás(n, x)  s ← 0  **Amíg** x[n] > 99 **végezd el**  x[n] ← x[n] **DIV** 10  **vége(amíg)**  s ← s + x[n]  **térítsd** s  **Vége(algoritmus)** |

**Megoldás**

Adott egy feladatszöveg és több algoritmus, amelyek közül ki kell választa­nunk a helyes algoritmust, esetleg algoritmusokat. Azt gondolhatnánk, hogy ez nem lehetséges csak úgy, ha a papíron „végrehajtjuk” mindegyiket. De erre most nincs szükség: ha figyelmesen tanulmányozzuk az algoritmusokat, észrevesszük, hogy nincs nagy különbség köztük, és el lehet különíteni a helyeseket anélkül, hogy táblázatban követnénk az értékeket.

Ha összehasonlítjuk az **A.** és **B.** algoritmusokat, azonnal észrevesszük, hogy az egyetlen különbség a két algoritmus első sorában van: az **A.** algoritmus ***s***-nek 0 kezdőértéket ad, míg a **B.** algoritmusban a kezdőérték ***n***. Természetesen, az ***n*** nincs mit keressen itt, hiszen ***n*** az ***x*** sorozat mérete! Így a **B.** algoritmust hibásnak minősítjük.

A továbbiakban az **A.** algoritmust bejárjuk gondolatban, és belátjuk, hogy helyes.

Mivel van egy jónak talált algoritmusunk, most ezt összehasonlítjuk a **C.** és **D.** algoritmusokkal. Észrevesszük, hogy a **C.**-ben az aktuális elem „túl korán” kerül be az összegbe, míg a **D.** algoritmus nem figyeli a sorozat végét, sőt az ***n*** változó értéke egyáltalán nem változik. Tehát a helyes válasz: **A**.

1. **Különleges számok sorozatának generálása (6 pont)**

Legyen az ***s*** egy természetes számokat tároló sorozat, ahol

***si*** = , (***i*** = 1, 2, ...). A művelet a bal és a jobb operandus számjegyeit konkatenálja (egymás után ragasztja) ebben a sorrendben, az ***x*** pe­dig egy természetes szám (1 ≤ ***x*** ≤ 99). Például, ha ***x*** = 3, az ***s*** sorozat elemei a következők: 3, 4, 43, 434, 43443, ... .

Állapítsátok meg, hány számjegye van az ***s*** sorozat azon elemének, amely a ***k*** számjegyű elem előtt található (1 ≤ ***k*** ≤ 30).

1. ha ***x*** = 15 és ***k*** = 6, az ***s*** sorozatban a ***k*** számjegyű elem előtt levő elem szám­jegyeinek száma 5.
2. ha ***x*** = 2 és ***k*** = 8, az ***s*** sorozatban a ***k*** számjegyű elem előtt levő elem számje­gyeinek száma 5.
3. ha ***x*** = 14 és ***k*** = 26, az ***s*** sorozatban a ***k*** számjegyű elem előtt levő elem számjegyeinek száma 16.
4. ha ***x*** = 5 és ***k*** = 13, az ***s*** sorozatban a ***k*** számjegyű elem előtt levő elem számjegyeinek száma 10.

**Megoldás**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **A.** | **B.** | **C.** | **D.** |
| ***Elem sorszáma*** | ***k*** | ***k*** | ***k*** | ***k*** |
| 1 | 2 | 1 | 2 | 1 |
| 2 | 2 | 1 | 2 | 1 |
| 3 | 4 ≠ 5 | 2 | 4 | 2 |
| 4 | **6** | 3 | 6 | 3 |
| 5 |  | 5 | 10 | 5 |
| 6 |  | **8** | 16 | 8 ≠ 10 |
| 7 |  |  | **26** | **13** |

Mivel a feladat nem kéri a számsorozatot – elég, ha az elemek hosszával fog­lalkozunk. Észrevesszük, hogy egy bizonyos elem hossza egyenlő a közvetlenül előtte levő két elem hosszának összegével (számjegyeiknek darabszámával).

Az **A.** válasz ***x*** = 15-ből indul, a ***k*** = 6 számjegyű elemre hivatkozik, és feltéte­lezi, hogy az ***s*** sorozatban a ***k*** számjegyű elem előtt levő elem számjegyeinek száma 5. Mivel a 6 számjegyű előtti szám négy számjegyű, az **A.** válasz nem he­lyes.

A **B.** válasz ***x*** = 2**-**ből indul, a ***k*** = 8 számjegyű elemre hivatkozik, és feltéte­lezi, hogy a sorozatban a ***k*** számjegyű elem előtt levő elem számjegyeinek száma 5. Mivel a 8 számjegyű előtti elem valóban ötszámjegyű, a **B.** válasz helyes.

A **C.** válasz 14-ből indul, a ***k*** = 26 számjegyű elemre hivatkozik, és feltételezi, hogy a ***k*** számjegyű elem előtt levő elem számjegyeinek száma 16, ami igaznak bizonyul (**C.** helyes).

A **D.** válasz 5-ből indul, a ***k*** = 13 számjegyű elemre hivatkozik, és feltételezi, hogy a ***k*** számjegyű elem előtt levő elem számjegyeinek száma 10, ami hamisnak bizonyul, tehát **D.** nem helyes.

1. **Körkörös permutációk (6 pont)**

Legyen az ***n*** (3 ≤ ***n*** ≤ 10 000) elemű ***x*** sorozat, amely természetes számokat tárol és a ***k*** természetes szám (1 ≤ ***k*** < ***n***). A körkörösPerm(n, k, x) algoritmus­nak az ***x*** sorozat körkörös permutációját kellene generálnia ***k*** pozícióval balra. Például, a (4, 5, 2, 1, 3) sorozat az (1, 3, 4, 5, 2) sorozat körkörös permutációja két pozícióval balra. Sajnos, a körkörösPerm(n, k, x) algoritmus nem helyes, mivel ***n*** és ***k*** bizonyos értékeire hibás eredményt generál.

|  |  |
| --- | --- |
| **Algoritmus** körkörösPerm(n, k, x)  c ← k  **Minden** j = 1, c **végezd el**  hova ← j; szám ← x[hova]  **Minden** i = 1, n / c – 1 **végezd el**  honnan ← hova + k  **Ha** honnan > n **akkor**  honnan ← honnan - n  **vége(ha)**  x[hova] ← x[honnan]  hova ← honnan  **vége(minden)**  x[hova] ← szám  **vége(minden)**  **Vége(algoritmus)** | Válasszátok ki ***n***, ***k*** és ***x*** értékeit, me­lyekre a körkörösPerm(n, k, x) algorit­mus az ***x*** sorozat körkörös permutációját gene­rál­ja, ***k*** pozícióval balra:   1. ***n*** = 6, ***k*** = 2, ***x*** = (1, 2, 3, 4, 5, 6) 2. ***n*** = 8, ***k*** = 3, ***x*** = (1, 2, 3, 4, 5, 6, 7, 8) 3. ***n*** = 5, ***k*** = 3, ***x*** = (1, 2, 3, 4, 5) 4. ***n*** = 8, ***k*** = 4, ***x*** = (1, 2, 3, 4, 5, 6, 7, 8) |

**Megoldás**

A legértékesebb észrevétel a bemeneti adatokra vonatkozik, ugyanis ***n*** és ***k*** értékei a **B.** és **C.** esetekben relatív prímek (8, 3; és 5, 3), az **A.** és **D.** esetekben pedig ***n*** -nek és ***k***-nak van közös osztója (6, 2; és 8, 4). Sejtjük, hogy ez lehet az adatok közti különbség, amelynek következtében egyes bemeneti adatokra nem működik helyesen az algoritmus. Készítünk táblázatot ***n*** = 6-ra és ***k*** = 2-re:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ***n*** | ***k*** | ***c*** | ***j*** | ***hova*** | ***szám*** | ***i*** | ***honnan*** | ***x*** |
|  | 6 | 2 |  |  |  |  |  |  | 1, 2, 3, 4, 5, 6 |
| **1. *bejárás* (*j =* 1)** |  |  | 2 | 1 | 1 | 1 | 1 | 3 | **3**, 2, 3, 4, 5, 6 |
|  |  |  | 1 | 3 |  | 2 | 5 | **3**, 2, **5**, 4, 5, 6 |
|  |  |  | 1 | 5 |  |  |  | **3**, 2, **5**, 4, **1**, 6 |
| **2. *bejárás* (*j =* 1)** |  |  |  | 2 | 2 | 2 | 1 | 4 | **3**, **4**, **5**, 4, **1**, 6 |
|  |  |  | 2 | 4 |  | 2 | 6 | **3**, **4**, **5**, **6**, **1**, 6 |
|  |  |  | 2 | 6 |  |  |  | **3**, **4**, **5**, **6**, **1**, **2** |

A fenti esetben az algoritmus helyesen működik. Lássuk mi történik, ha ***n*** = 5 és ***k*** = 3? Mivel ***c*** = ***k*** és az ***i*** által vezérelt **Minden** ciklus 1-től (***n* / *c* – 1**)-ig dolgozik, ahol a végsőérték a példánk esetében 5 / 3 – 1 = 0, a ciklus törzsét az algoritmus egyszer sem hajtja végre. Ennek következtében a sorozat nem válto­zik. Tehát a **C.** válasz nem helyes. Innen tovább, végre lehetne hajtani az algorit­must a **B.** és a **D.** válaszok esetében is. De kiindulhatunk mindkét esetben a fen­tiekből, vagyis **A.** és **D.** helyes, **B.** és **C.** nem helyes.

1. **Vajon mit csinál? (6 pont)**

Az ***m*** és ***n*** hosszú oldalakkal rendelkező téglalap fel van osztva 1 oldalhosszú­ságú négyzetecskékre (***m***, ***n*** – természetes számok, 0 < ***m*** < 101, 0 < ***n*** < 101). Adott a téglalap(m, n) algoritmus. Mi a hatása ennek az algoritmusnak?

|  |  |
| --- | --- |
| **Algoritmus** téglalap(m, n)  d ← m  c ← n  **Amíg** d ≠ c **végezd el**  **Ha** d > c **akkor**  d ← d – c  **különben**  c ← c – d  **vége(ha)**  **vége(amíg)**  **térítsd** m + n – d  **Vége(algoritmus)** | Kiszámítja és téríti azoknak az 1 oldalhosszú­ságú négyzetecskéknek a számát, amelyeket át­vág a téglalap egy átlója.  Meghatározza ***d***-ben a téglalap oldalainak leg­nagyobb közös osztóját és téríti az oldalak összegének és ***d***-nek a különbségét.  Ha ***m*** = 8 és ***n*** = 12, a térített érték 16.  Ha ***m*** = 6 és ***n*** = 11, a térített érték 15. |

**Megoldás**

A **B.** válasz biztos nem jó, hiszen „az oldalak összege” mindkét oldalt kétszer kellene tartalmazza. Könnyen belátjuk, hogy a **C.** helyes (8 + 12 – 4 = 16), de a **D.** nem (6 + 11 – 1 = 16 ≠ 15). A legérdekesebb az **A.** válasz. A feladat nem kéri, hogy bebizonyítsuk az állítás helyességét, itt mégis leírjuk a gondolatmenetet, amelynek alapján eldönthető, hogy az **A.** válasz is helyes.

Előbb megvizsgáljuk azt az esetet, amikor lnko(*m*, *n*) = 1. Ekkor az átló nem tartalmaz egyetlen rácspontot sem a végpontokon kívül (és így könnyű megszá­molni az átvágott kis négyzeteket).

*P*(*m*, *n*)

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

Valóban, ha a téglalapot egy olyan koordináta-rendszerbe helyezzük, amely­ben *O*(0, 0) a téglalap bal alsó sarka és *P*(*m*, *n*) a jobb felső sarka, akkor az *OP* egyenes egyenlete *y* = . Mivel az tört irreducibilis, ezért az egyetlen *x* ∈ {1, 2, ..., *m* – 1} értékre sem lesz egész szám, ami pontosan azt jelenti, hogy nincs több rácspont az *OP* szakaszon a végpontokon kívül (és ez hasonlóan iga­zolható a másik átlóra is).

A bal alsó, *O* sarokkal rendelkező kis négyzetből indulunk a jobb felső, *P* sa­rokkal rendelkező kis négyzet felé úgy, hogy minden lépésben vagy felfele, vagy jobbra lépünk. Ahhoz, hogy megszámoljuk az *OP* átló által átvágott kis négyze­teket, meg kell számolnunk, hogy hány kis négyzetet érintünk az út során. Ez a szám egyenlő *m* + *n* – 1-gyel.

Általános esetben legyen *lnko*(*m*, *n*) = *d*, tehát létezik *m*1, *n*1 ∈ **N** úgy, hogy *m* = *m*1 \* *d* és *n* = *n*1 \* *d*, valamint *lnko*(*m*1, *n*1) = 1. Ez viszont azt jelenti, hogy az *m* × *n*-es téglalap felbomlik kis *m*1 × *n*1-es téglalapokra, az alábbi ábra szerint.

*P*(*m*, *n*) = *Pd*(*dm*1, *dn*1)

*P*1(*m*1, *n*1)

*Pd* – 1((*d* – 1)*m*1, (*d –* 1)*n*1)

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
| *O*(0, 0) |  |  |  |

Könnyen észrevehető, hogy

* minden sorban és minden oszlopban *d* darab *m*1 × *n*1-es téglalap van;
* a főátló mentén összesen *d* darab kis téglalap helyezkedik el, és ezeknek a megfelelő csúcsai is a főátlón vannak (ez kijön a következő hasonló­ságból: , tehát a *P*1(*m*1, *n*1) pont is a nagy téglalap főátló­ján van, hasonlóan a *P*2(2\**m*1, 2\**n*1) is stb.)

A fentiek alapján az általános esetben az *OP* főátló annyi kis négyzetet vág ketté, mint amennyit a főátlón elhelyezkedő kisebb, *m*1 × *n*1-es téglalapokban kettévág összesen, vagyis pontosan *d* \* (*m*1 + *n*1 –1) = *m* + *n* – *d* darabot.

1. **Dominók elhelyezése az átlóra (6 pont)**

Legyen egy téglalap alakú tábla, amely föl van osztva ***n***×***m*** cellára (***n*** – a sorok száma és 2 ≤ ***n*** ≤ 100, ***m*** – az oszlopok száma és 2 ≤ ***m*** ≤ 100, ahol ***n*** és ***m*** természetes számok). Két játékos, A és B, felváltva lépéseket hajtanak végre: minden lépésnél a soron levő játékos megjelöl egyetlen cellát, amely átlósan szomszédos az előző lépésben, a másik játékos által megjelölt cellával és amely még nincs megjelölve. Az a játékos, akinek nincs hova lépnie, veszít. Az A játé­kos lép először, és megjelöl egy cellát a táblán. Határozzátok meg, milyen felté­telek mellett van A-nak biztos nyerési stratégiája, (vagyis nyerni fog, B lépéseitől függetlenül) és mi lehet A első lépése ahhoz, hogy nyerjen.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | A |  |  |  |  | A |  |  |  |  | A |  |  |  |  | A |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  | B |  |  |  |  | B |  |  |  |  | B |  | B |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | A |  |  |  |  | A |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | a |  |  |  |  | b |  |  |  |  | c |  |  |  |  | d |  |  |  |  | e |  |  |

Példa: a) eredeti állapot (*n* = 5 és *m* = 4), b) az első lépés utáni helyzet (lépett A), c) a második lépés után (lépett B), d) a harmadik lépés után (lépett A), e) a negyedik lépés után (lépett B)

1. feltétel: ***m*** páratlan szám;

az A játékos először a fenti első sorban (1-es sor) és egy páratlan sorszámú oszlopban levő cellára lép.

1. feltétel: ***n*** páratlan szám;

az A játékos először egy páros sorszámú sorban és a tábla bal első oszlopában (1-es oszlop) levő cellára lép.

1. feltétel: mindkét szám (***n*** és ***m***) páros szám;

az A játékos először a tábla bal felső sarkában (1-es sor és 1-es oszlop) levő cellára lép.

1. feltétel: ***n*** és ***m*** közül legalább az egyik páratlan szám;

az A játékos először a tábla bal felső sarkában (1-es sor és 1-es oszlop) levő cellára lép.

**Megoldás**

Könnyű belátni, hogy legalább egy számnak az ***m*** és ***n*** közül páratlannak kell lennie, ahhoz, hogy az „ellenfél” leszorítható legyen a tábláról egy adott pillanat­ban. Így a **C.** választ helytelennek minősítjük. Mivel a lépések átlósan szomszé­dos cellára történnek, az **A.** válasz biztos helyes. Ahhoz, hogy vizsgálhassuk, mi történik a **D.** esetben, feltételezzük, hogy ***n*** páratlan és ***m*** páros. Ha B átlósan lefele lép, egy adott pillanatban a szélre ér, de A léphet az előzőtől átlósan lefele. De ez fordítva is lehetséges. Ugyanígy ellenőrizhetnénk a többi lépést is, de egy­szerű számolással is eljuthatunk a következtetésre, hogy **A.** és **D.** helyes válaszok.

**Kalitkák**

Az állatkertben a papagájok 1-től ***n***-ig számozott kalitkák­ban élnek (1 ≤ ***n*** ≤ 10 000). Egy adott pillanatban egy játékos majom kinyit minden kalitkát. Megijed a következmények­től, visszatér az első kalitkához és bezár minden második kalitkát (így bezárja a 2, 4, 6, ... sorszámúakat). A majomnak megtetszik ez a játék. Ezért újra elindul az elejéről és meg­látogat minden harmadik kalitkát (vagyis a 3, 6, 9, ... sorszámúakat) és bezárja a kalitkát, ha az nyitva van, illetve kinyitja, ha azt zárva találja. A negyedik bejá­ráskor meglátogat minden negyedik kalitkát, és hasonlóan jár el (megváltoztatva a meglátogatott kalitka állapotát). A majom megismétli a játékot, míg az utolsó bejáráskor (az ***n***. bejárás) bezárja az ***n***. kalitkát, ha ez nyitva van vagy kinyitja, ha zárva van.

**Követelmények:**

1. Hány kalitka marad nyitva az utolsó bejárás után, ha ***n*** = 10? (2 pont)
2. Mely sorszámú kalitkák maradnak nyitva az utolsó bejárás után, ha ***n*** = 10? (2 pont)
3. Összesen hányszor látogatja meg a majom a ***k*** sorszámú kalitkát (1 ≤ ***k*** ≤ ***n***) az ***n*** bejárás során? Indokoljátok meg a választ. (4 pont)
4. Mi annak szükséges és elégséges feltétele, hogy a ***k*** sorszámú kalitka (1 ≤ ***k*** ≤ ***n***) nyitva maradjon az ***n*** kalitka utolsó bejárása után? Indokoljátok meg a választ. (4 pont)
5. Hány kalitka marad nyitva az ***n*** kalitka utolsó bejárása után? Indokoljátok meg a választ. (4 pont)
6. Írjatok algoritmust, amely kiszámítja az utolsó bejárás után *nyitva maradt kalitkák számát* (***nyitvaSz***). Az algoritmus bemeneti paramétere a kalitkák ***n*** (1 ≤ ***n*** ≤ 10 000) száma, kimeneti paramétere a ***nyitvaSz*** szám. (14 pont)

**1. *Példa:*** ha ***n*** = 5, akkor ***nyitvaSz*** = 2 (nyitva marad az 1-es és a 4-es sorszámú kalitka).

**2*. Példa:*** ha ***n*** = 12, akkor ***nyitvaSz*** = 3.

**Megoldás**

A majom egy bejárás alkalmával, akkor és csakis akkor látogat meg egy kalit­kát, ha a kalitka sorszáma a bejárás sorszámának többszöröse. Következik, hogy egy adott kalitka meglátogatásának darabszáma egyenlő a kalitka sorszáma kü­lönböző osztóinak darabszámával. A kalitka akkor és csakis akkor marad nyitva, ha a sorszámának *páratlan* darabszámú osztója van.

Lássuk, mi történik, ha ***n*** = 5. Jelöljük a ***bejárásSorszáma*** változóval a kalitka­sor aktuális bejárásának sorszámát. Ha a kalitkák száma ***n***, a ***bejárásSorszáma*** az {1, 2, ..., ***n***} halmazból kap értéket.

Kövessük a példát:

1. Az első bejáráskor (***bejárásSorszáma*** = 1) a majom kinyitja az 1, 2, 3, 4, 5 sorszámú kalitkákat. Tehát most nyitva vannak az **1 2 3 4 5** sorszámú kalitkák. Észrevesszük, hogy az 1, 2, 3, 4, 5 számoknak ***egyetlen*** osztójuk van az {1, ..., ***bejárásSorszáma***} halmazban, ami most {1}. Így az egyetlen osztó az 1.
2. A második bejárás során (***bejárásSorszáma*** = 2) a majom bezár minden má­sodik kalitkát, vagyis a 2 és a 4 sorszámúakat. Nyitva maradnak az **1 3 5** sor­számúak, ahol mindegyiknek ***egyetlen*** osztója van az {1, ..., ***bejárásSorszáma***} = {1, 2} halmazban éspedig az 1, miközben a 2-es és a 4-es sorszámúak zárva vannak. Ennek a két számnak egyenként ***két*** osztója van az {1, 2} halmazban, éspedig az 1 és a 2.
3. A harmadik bejárás során (***bejárásSorszáma*** = 3) a majom kinyit minden har­madik kalitkát, ha ez zárva van és bezár minden harmadik kalitkát, ha ez nyit­va van. Így bezárja a 3. kalitkát, miközben nyitva maradnak: **1** és **5**. Az 1-nek és az 5-nek ***egyetlen*** osztója van az {1, ..., ***bejárásSorszáma***} = {1, 2, 3} hal­mazban, miközben a 2, 3, 4 sorszámú kalitkák zárva vannak. Észrevesszük, hogy ezeknek a számoknak egyenként ***két*** osztója van az {1, 2, 3} halmazban: 2 osztói az 1 és a 2, a 3 osztói az 1 és a 3, a 4 osztói pedig az 1 és a 2).
4. A negyedik bejárás során (***bejárásSorszáma*** = 4) a majom kinyitja a 4-es sor­számú kalitkát (zárva volt). Most nyitva vannak az **1**, **4** és **5** sorszámú kalitkák, ahol 1-nek egy osztója van az {1, 2, 3, 4} halmazban, 4-nek három osztója van (1, 2 és 4), 5-nek pedig egy osztója van (***az osztók száma páratlan***). A 2-es és 3-as kalitkák zárva vannak (2-nek és 3-nak két-két osztója van a megfelelő halmazban).
5. Az ötödik bejárás (***bejárásSorszáma*** = 5) végén a majom bezárja az ötödik kalitkát. Nyitva vannak az **1** és a **4** (***páratlan*** számú osztóik vannak az {1, 2, 3, 4, 5} halmazban) és zárva vannak a 2-es, 3-as és 5-ös kalitkák (ahol 2-nek, 3-nak és 5-nek ***két-két*** osztója van).

**Állítás**

Egy természetes számnakakkor és csakis akkor van páratlan számú osztója, ha ***négyzetszám***.

**Bizonyítás**

Ha ***d*** az ***m*** természetes szám osztója, akkor ***m*** / ***d*** is osztója ***m***-nek, mivel:

***d*** \* (***m*** / ***d***) = ***m*** (1)

Következik, hogy egy természetes szám osztói párba állíthatók, vagyis úgy gondolhatnánk, hogy bármely természetes számnak páros darabszámú osztója van. Másfelől, ha van olyan pár, amelyben a két szám egyenlő, következik, hogy az ***m*** szám négyzetszám, mivel, ha behelyettesítjük a ***d***-t (***m*** / ***d***)-vel az (1)-es egyenletben, kapjuk, hogy ***d***\* ***d*** = ***m***. Ezekben az esetekben, csökkentjük a páros számú osztók darabszámát, amiből következik, hogy az osztók száma páratlan.

Ha ***d*** ≠ ***m*** / ***d***, marad a páros darabszámú osztószám (a nem négyzetszámok esetében).

Ilyen körülmények között a feladat megoldása visszavezethető azoknak a ka­litkáknak a megszámolására, amelyeknek sorszáma négyzetszám. Egy ***n*** termé­szetes szám négyzetszám, ha \* = ***n***. Másfelől *az n-nél kisebb (vagy egyenlő) négyzetszámok darabszáma egyenlő* ⎦-nel. Ez az érték meghatározható a *bi­náris keresés módszerével*, amikor keressük a -et az 1, 2, ..., ***n***/2 rendezett számsorozatban.

A kalitkákat „zárhatjuk/nyithatjuk” a következő táblázatban:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Bejárás­sorszáma*** | ***Kalitkák*** | | | | | ***Osztók halmaza*** |
| 1 | 2 | 3 | 4 | 5 |  |
| kiindulópont | zárva | zárva | zárva | zárva | zárva |  |
| 1 | nyitva | nyitva | nyitva | nyitva | nyitva | 1 |
| 2 | nyitva | zárva | nyitva | zárva | nyitva | 1, 2 |
| 3 | nyitva | zárva | zárva | zárva | nyitva | 1, 2, 3 |
| 4 | nyitva | zárva | zárva | nyitva | nyitva | 1, 2, 3, 4 |
| 5 | **nyitva** | zárva | zárva | **nyitva** | zárva | 1, 2, 3, 4, 5 |

Lássuk a válaszokat a feladatban feltett kérdésekre:

1. Ha ***n*** = 10, nyitva marad 3 kalitka.
2. Ha ***n*** = 10, nyitva maradnak az 1, 4, 9 sorszámú kalitkák.
3. A ***k*** sorszámú kalitkát összesen annyiszor látogatja meg a majom az ***n*** be­járás során, ahány osztója van ***k***-nak az {1, 2, ..., ***n***} halmazban.
4. Egy ***k*** sorszámú kalitka akkor és csakis akkor marad nyitva az ***n*** kalitka utolsó bejárása után, ha:

* ***k***-nak *páratlan darabszámú osztója* van az {1, 2, ..., ***n***} halmazban

vagy – másképp kifejezve:

* ***k*** *négyzetszám*, mivel minden olyan szám, amelynek páratlan darab­számú osztója van négyzetszám.

1. A nyitva maradt kalitkák darabszáma egyenlő-nel, mivel minden olyan szám, amelynek páratlan darabszámú osztója van, négyzetszám. Ugyanakkor az ***n***-nél kisebb négyzetszámok darabszáma:  .
2. Az algoritmus több elképzelés alapján is megtervezhető. A következőkben bemutatjuk az egyik legkézenfekvőbbet (**V1**), majd egy hatékonyabbat (**V2**).
3. a nyitva maradt kalitkák számát meghatározzuk az ***n***-nél kisebb négyzetszámok megszámlálása által (egyetlen ismétlő struktú­rával);
4. a nyitva maradt kalitkák számát a  alapján határozzuk meg; az optimális algoritmus ezt az értéket a bináris keresés algoritmusával számítja ki:

|  |  |
| --- | --- |
| **int** kalitkakV1(**int** n){  **int** nyitvaSz = 0;  **for** (**int** i = 1; i \* i <= n; i++){  nyitvaSz++;  }  **return** nyitvaSz;  } | **int** kalitkakV2(**int** n){  **int** bal = 1;  **int** jobb = n / 2;  **while** (bal < jobb){  **int** k = ((bal + jobb) / 2);  **if** (k \* k >= n) jobb = k;  **else** bal = k + 1;  }  **if** (bal \* bal <= n)  **return** bal;  **else**  **return** bal - 1;  } |

1. ha a fenti ötletek közül egyik sem „ugrik be”, szimulálhatjuk a ma­jom tevékeny­sé­gét: bejárunk minden kalitkát ***n***-szer, kinyitjuk/be­zárjuk a kalitkát a bejárás során, majd a nyitva maradt kalitkákat megszámláljuk.

|  |
| --- |
| **int** kalitkakV3(**int** n){  **const** **int** MAX = 10001;  **bool** ajtok[MAX];  **for** (**int** i = 1; i <= n; i++)  ajtok[i] = **true**;  **for** (**int** k = 2; k <= n; k++){  **for** (**int** i = k; i <= n; i = i + k){  **if** (ajtok[i]) ajtok[i] = **false**;  **else** ajtok[i] = **true**;  }  }  **int** nyitvaSz = 0;  **for** (**int** i = 1; i <= n; i++)  **if** (ajtok[i])  nyitvaSz++;  **return** nyitvaSz;  } |