**Válogattam egy Mat–Infó Verseny feladatai közül (kihagytam azokat, amelyek a rekurzió ismeretét kérik; rekurzióval február 13.-án foglalkozunk).**

**Oldjátok meg ezeket, a helyes válaszokat megbeszéljük a konzultáción.**

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**. | | |

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

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)** |

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.
5. **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) |

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. |

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.

**B rész (30 pont)**

**Oldjátok meg ezt a feladatot is, még akkor is, ha algoritmust nem kell írnotok majd.**

**Ebből a feladatból is lehet csinálni jó nehéz rácstesztet 😊**

**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.