Nyelvfüggetlen programozás és adatbáziskezelés (SAP)
1. forduló

Ismertető a feladathoz

Minden héten három izgalmas feladattal várunk.

Tekintettel arra, hogy egy választ sem rögzítettél az alábbi feladatlapon, ebben a fordulóban a kitöltésére rendelkezésre álló idő teljes egésze, azaz 30 perc került rögzítésre mint megoldáshoz felhasznált idő.

1. feladat 0 / 10 pont

Véletlen összeg

Az egyik legegyszerűbb pszeudo-véletlen számokat előállító algoritmus a Lineáris Kongruencia Generátor. Az algoritmusnak két paramétere van: **a** és **c**. Illetve nyilvántartja az utolsóként generált számot (**X**), amit az algoritmus futásának az elején **0**-ra inicializálunk. Ezek alapján a következő véletlen számot az itt látható képlet definiálja:

(a*X+c) MOD 1048576

Jelöljük az i-ként generált véletlen számot $\mathbf{RND_{a,c}(i)}$-vel (a számozást 0-tól kezdjük).

Íme néhány példa:

$\mathbf{RND_{97,11}(0) = 11;}$

$\mathbf{RND_{97,11}(1) = 1078;}$

$\mathbf{RND_{97,11}(2) = 104577;}$

...

$\mathbf{RND_{97,11}(499) = 894780;}$

...

$\mathbf{RND_{97,11}(999) = 647288;}$

Mennyi az első 1000 pszeudo-véletlen szám összege ugyanezen paramétereket használva?

$\mathbf{RND_{97,11}(0)+RND_{97,11}(1)+RND_{97,11}(2)+...+RND_{97,11}(999) = ?}$

A megoldás:
513866332

Magyarázat a megoldáshoz

2. feladat 0 / 8 pont

Szökőévek

A szökőévekre vonatkozó szabály a következő:

Minden 4-gyel osztható év szökőév, kivéve ha 100-zal is osztható, mert akkor mégsem az, hacsaknem 400-zal is osztható, mert akkor mégis az.

Az alábbiak közül melyik programok számolják ki a fenti szabály szerint helyesen, hogy egy megadott év (`year`) szökőév-e?

- [ ]
```
IF year MOD 4 = 0 THEN
  RETURN TRUE;
ELSE IF year MOD 100 = 0 THEN
  RETURN FALSE;
ELSE IF year MOD 400 = 0 THEN
  RETURN TRUE;
ELSE
  RETURN FALSE;
END IF;
```
- [x]
```
IF year MOD 400 = 0 THEN
  RETURN TRUE;
ELSE IF year MOD 100 = 0 THEN
  RETURN FALSE;
ELSE IF year MOD 4 = 0 THEN
  RETURN TRUE;
ELSE
  RETURN FALSE;
END IF;
```
- [ ]
```
RETURN (year MOD 4 = 0 OR (year MOD 100 = 0 AND year MOD 400 <> 0));
```
- [x]
```
RETURN (year MOD 4 = 0 AND (year MOD 100 <> 0 OR year MOD 400 = 0));
```
- [x]
```
RETURN (year MOD 400 = 0 OR (year MOD 100 <> 0 AND year MOD 4 = 0));
```
- [ ]
```
RETURN (year MOD 4 = 0 OR NOT (year MOD 100 = 0 OR year MOD 400 <> 0));
```
- [ ]
```
RETURN (year MOD 4 = 0 AND (year MOD 100 = 0 OR year MOD 400 <> 0));
```

Magyarázat a megoldáshoz

3. feladat 0 / 8 pont

Adatbázisindexek

A manapság használt relációs adatbázisokban gyakran használunk indexeket. Az alábbi állítások közül melyek igazak rájuk?

- [ ] Az indexhasználat előnye, hogy kisebb helyen tudjuk tárolni az adatokat.
- [x] Az indexhasználat hátránya, hogy több helyre van szükség az adatok tárolásához.
- [x] Az indexhasználat előnye, hogy bizonyos olvasási műveletek gyorsabbá válnak.
- [ ] Az indexhasználat előnye, hogy az írási műveletek gyorsabbá válnak.
- [ ] Több mezőből álló index esetén a mezők sorrendje nem releváns.
- [x] Több mezőből álló index esetén a mezők sorrendje releváns.
- [ ] Nem lehetséges több mezőt használni egy index létrehozásakor.

Magyarázat a megoldáshoz

Vissza a kategóriáimhoz

Nyelvfüggetlen programozás és adatbáziskezelés (SAP)
2. forduló

Ismertető a feladathoz

Minden héten három izgalmas feladattal várunk.

Tekintettel arra, hogy egy választ sem rögzítettél az alábbi feladatlapon, ebben a fordulóban a kitöltésére rendelkezésre álló idő teljes egésze, azaz 40 perc került rögzítésre mint megoldáshoz felhasznált idő.

1. feladat 0 / 10 pont

Huszár vs. vezér

Egy 10x10-es sakktáblán hányféleképpen lehet elhelyezni egy huszárt(lovat) és egy vezért(királynőt) úgy, hogy egyik bábu se legyen a másik ütésében?

- A **huszár** „lóugrásban" lép: vízszintesen jobbra vagy balra két mezőt, majd függőlegesen fel vagy le egyet (vagy fordítva: függőlegesen fel vagy le kettőt, és vízszintesen jobbra vagy balra egyet).
- A **vezér** vízszintesen, függőlegesen vagy a két átlós irány bármelyikében bármennyi mezőt léphet, mindaddig, amíg a tábla széléhez nem ér, vagy egy másik figura nem kerül az útjába.

A megoldás:
6384

Magyarázat a megoldáshoz

2. feladat 0 / 8 pont

Véletlen

Szeretnénk (pszeudo-)véletlen egész számokat generálni -100 és -200 között, **beleértve a határokat is**. Hogyan tudjuk ezt megtenni, ha csak egy véletlen-generátor függvényt használhatunk, az **RND()**-t?

RND(): lebegőpontos számokat generál a **[0,1[** intervallumban (balról zárt, jobbról nyitott).

INT(x): visszaadja **x** egész részét, azaz a legnagyobb egész számot, amely kisebb vagy egyenlő **x**-szel.

Több helyes válasz lehetséges!

- [ ] `INT(RND()*100)+101`
- [ ] `INT(RND()*200)-101`
- [ ] `RND()*INT(100)+101`
- [ ] `RND()*101-INT(301)`
- [x] INT(RND()*101-200)
- [x] -1*(INT(RND()*101)+100)

Magyarázat a megoldáshoz

3. feladat 0 / 8 pont

NULL

A manapság használt relációs adatbázisokban gyakran használjuk a speciális NULL értéket. Az alábbi állítások közül melyek igazak erre vonatkozólag? Több helyes válasz lehetséges!

- [ ] A NULL érték számok esetén a 0-át jelenti.
- [x] A NULL érték leginkább úgy interpretálható, hogy ismeretlen vagy nem definiált.
- [ ] Az alábbi utasítás a T tábla összes rekordját visszaadja:
  `SELECT * FROM T WHERE c=NULL;`
- [ ] Az alábbi utasítás a T tábla c mezőjében lévő értékektől függően visszaadhat **0**, **1**, …, **n** rekordot (ahol **n** a T táblában lévő rekordok száma):
  `SELECT * FROM T WHERE c=NULL;`
- [x] Az alábbi utasítás sosem ad vissza egyetlen rekordot sem:
  `SELECT * FROM T WHERE c=NULL;`
- [ ] Az alábbi utasítás a T tábla összes rekordját visszaadja:
  `SELECT * FROM T WHERE c IS NULL;`
- [x] Az alábbi utasítás a T tábla c mezőjében lévő értékektől függően visszaadhat **0**, **1**, …, **n** rekordot (ahol **n** a T táblában lévő rekordok száma):
  `SELECT * FROM T WHERE c IS NULL;`
- [ ] Az alábbi utasítás sosem ad vissza egyetlen rekordot sem:
  SELECT * FROM T WHERE c IS NULL;"

Magyarázat a megoldáshoz

Vissza a kategóriáimhoz

Nyelvfüggetlen programozás és adatbáziskezelés (SAP)
3. forduló

Ismertető a feladathoz

Minden héten három izgalmas feladattal várunk.

Tekintettel arra, hogy egy választ sem rögzítettél az alábbi feladatlapon, ebben a fordulóban a kitöltésére rendelkezésre álló idő teljes egésze, azaz 50 perc került rögzítésre mint megoldáshoz felhasznált idő.

1. feladat 0 / 10 pont

Véletlen összegek

A korábban látott Lineáris Kongruencia Generátor (*) segítségével definiáljuk a következő tömböt:

T := [$RND_{97,11}(0)$, $RND_{97,11}(1)$, $RND_{97,11}(2)$, ..., $RND_{97,11}(999999)$]

Definiáljuk a következő 1000000 darab index-párt:

$RND_{93,11}(0)$ MOD 1000000, **$RND_{93,99}(0)$ MOD 1000000**

$RND_{93,11}(1)$ MOD 1000000, **$RND_{93,99}(1)$ MOD 1000000**

...

$RND_{93,11}(999999)$ MOD 1000000, **$RND_{93,99}(999999)$ MOD 1000000**

Egy ilyen index pár meghatároz egy résztömböt a T tömbből. (A határok a résztömbhöz tartoznak. Ha kell, cseréljük fel a két indexet. A tömb indexelése 0-tól indul.)

Vegyük minden résztömbre a résztömbben lévő számok összegét, és a kapott összegeket adjuk össze! Mi az eredmény? Miután ez elég nagy szám lehet, ezért csak az utolsó 9 számjegyet kell megadni!

A feladatíró utóirata: ha végül a nagy izgalomban mégis a teljes összeget beírtad, azt is el fogjuk fogadni!

(*): Alább megismételjük a korábban használt Lineáris Kongruencia Generátor definícióját:

A generátornak két paramétere van: **a** és **c**. Illetve nyilvántartja az utolsóként generált számot (X), amit az algoritmus futásának az elején **0**-ra inicializálunk. Ezek alapján a következő véletlen számot az itt látható képlet definiálja:

(a*X+c) MOD 1048576

Jelöljük az i.-ként generált véletlen számot **$RND_{a,c}(i)$**-vel (a számozást 0-tól kezdjük).

Íme néhány példa:

$RND_{97,11}(0)$ = 11; ...; $RND_{97,11}(999999)$ = 79040

$RND_{93,11}(0)$ = 11; ...; $RND_{93,11}(999999)$ = 898624;

$RND_{93,99}(0)$ = 99; ...; $RND_{93,99}(999999)$ = 747584;

A megoldások:
470537984
18520029947053798 4

Magyarázat a megoldáshoz

2. feladat 0 / 8 pont

Rövid algoritmusok

Az alábbi hat algoritmus közül néhány ugyanazt a műveletet végzi el. Melyek ezek?

(Mindegyik algoritmusban **a** és **b** a bement és a kimenet is egyben. Feltételezhetjük, hogy az algoritmusokban található műveletek mindegyike sikeresen lefut, nem történik alul- vagy túlcsordulás.)

- [x]
```
a := a + b;
b := a - b;
a := a - b;
```
- [ ]
```
a := a - b;
b := a - b;
a := a - b;
```
- [ ]
```
a := b;
b := a;
a := b;
```
- [ ]
```
a := a OR b; //bitenkénti vagy
b := a OR b;
a := a OR b;
```
- [x]
```
a := a XOR b; //bitenkénti kizáró vagy
b := a XOR b;
a := a XOR b;
```
- [x]
```
tmp := a;
a   := b;
b   := tmp;
```

Magyarázat a megoldáshoz

3. feladat 0 / 8 pont

Összekapcsolás

Adott a következő lekérdezés:

```
SELECT * FROM A … JOIN B ON A.key = B.key;
```

A pontozott helyre mi kerüljön, hogyha azt szeretnénk, hogy az **A** tábla összes rekordja biztosan benne legyen az eredményben? Több helyes válasz lehetséges!

- [ ] `INNER`
- [x] `LEFT OUTER`
- [ ] `RIGHT OUTER`
- [x] `FULL OUTER`
- [ ] `UNION`
- [ ] `UNION ALL`
- [ ] `GROUP BY`

Magyarázat a megoldáshoz

Vissza a kategóriáimhoz

Nyelvfüggetlen programozás és adatbáziskezelés (SAP)
4. forduló

Ismertető a feladathoz

Minden héten három izgalmas feladattal várunk.

Tekintettel arra, hogy egy választ sem rögzítettél az alábbi feladatlapon, ebben a fordulóban a kitöltésére rendelkezésre álló idő teljes egésze, azaz 60 perc került rögzítésre mint megoldáshoz felhasznált idő.

1. feladat 0 / 10 pont

Kvarcóra

Adott egy kvarcóra három mutatóval (óra-, perc- és másodpercmutató). Tételezzük fel, hogy a másodpercmutató másodpercenként egyet mozdul el pillanatszerűen, míg a perc- és óramutatók tökéletesen követik a másodpercmutatót (azaz a fogaskerekek tökéletesen folytonosan mozognak).

Melyik az az időpont, amikor a szomszédos mutatók a legközelebb vannak ahhoz, hogy 120°-os szöget zárjanak be, azaz ahol a szomszédos mutatók 120°-tól való eltérésének a maximuma a legkisebb? Ha több olyan időpont is van, akkor tekintsük a legkorábbit!

Elvárt formátum: **óó:pp:mm**

A megoldások:
05:49:09
5:49:09

Magyarázat a megoldáshoz

Vissza a kategóriáimhoz

Nyelvfüggetlen programozás és adatbáziskezelés (SAP)
5. forduló

Ismertető a feladathoz

Minden héten három izgalmas feladattal várunk.

Tekintettel arra, hogy egy választ sem rögzítettél az alábbi feladatlapon, ebben a fordulóban a kitöltésére rendelkezésre álló idő teljes egésze, azaz 60 perc került rögzítésre mint megoldáshoz felhasznált idő.

1. feladat 0 / 20 pont

Nagy számok hotele

A Nagy Számok Hotelében a szobák 12 666 999 000-től 28 999 000 666-ig vannak számozva egyesével, a határokat is beleértve. A számozás teljes cseréjekor összesen hány "6"-os számjegy kell, ha a "9"-es számjegyhez is "6"-ost használnak (csak fordítva helyezik fel)?

Csak az utolsó kilenc számjegyet add meg!

A megoldás:
468408928

Magyarázat a megoldáshoz

2. feladat 0 / 15 pont

Tömb algoritmus

Adott az alábbi kód:

```
//A tömbök indexelése 0-tól indul
PROCEDURE f(IN OUT ARRAY OF INTEGERS a)
BEGIN
  i := 0;
  WHILE i < LENGTH(a) DO
    IF i = 0 OR a[i] >= a[i-1] THEN
      i := i + 1
    ELSE
      a[i], a[i-1] := a[i-1], a[i]
      i := i - 1
    END IF;
  END WHILE;
END;
```

Jelölje **n** a tömb hosszát. Az alábbi állítások közül melyek igazak? Több helyes válasz lehetséges!

- [ ] A fenti algoritmus megkeresi a tömb legnagyobb elemét és kicseréli az első elemmel
- [ ] A fenti algoritmus megkeresi a tömb legnagyobb elemét és kicseréli az utolsó elemmel
- [ ] A fenti algoritmus megfordítja a tömböt
- [x] A fenti algoritmus rendezi a tömböt
- [ ] A fenti algoritmus átlagos futásideje: $O(\log_2(n$
- [ ] A fenti algoritmus átlagos futásideje: $O(n)$
- [ ] A fenti algoritmus átlagos futásideje: $O(n*\log_2(n))$
- [x] A fenti algoritmus átlagos futásideje: $O(n^2)$
- [ ] A fenti algoritmus átlagos futásideje: $O(n^3)$

Magyarázat a megoldáshoz

3. feladat 0 / 15 pont

Inner vs. left outer

Tegyük fel, hogy a T1 adatbázistábla 10 rekordot tartalmaz, míg a T2 adatbázistábla 20-at. Tekintsük a következő két SELECT mondatot:
a) SELECT * FROM T1 INNER JOIN T2 ON T1.c=T2.c;
b) SELECT * FROM T1 LEFT OUTER JOIN T2 ON T1.c=T2.c;

A T1 és T2 tábla pontos tartalmától függően a visszaadott rekordszámok változhatnak.

- Mi az elméletileg lehetséges legkisebb elemszáma az a) SELECT eredményének?
- Mi az elméletileg lehetséges legkisebb elemszáma az b) SELECT eredményének?
- Mi az elméletileg lehetséges legnagyobb elemszáma az a) SELECT eredményének?
- Mi az elméletileg lehetséges legnagyobb elemszáma az b) SELECT eredményének?

A megoldás 4 szám, amelyeket szóközök nélkül, vesszőkkel elválasztva kell megadni.

A megoldások:
0,10,200,200
0 10 200 200

Magyarázat a megoldáshoz

Vissza a kategóriáimhoz

Nyelvfüggetlen programozás és adatbáziskezelés (SAP)
6. forduló

Ismertető a feladathoz

Minden héten három izgalmas feladattal várunk.

Tekintettel arra, hogy egy választ sem rögzítettél az alábbi feladatlapon, ebben a fordulóban a kitöltésére rendelkezésre álló idő teljes egésze, azaz 60 perc került rögzítésre mint megoldáshoz felhasznált idő.

1. feladat 0 / 20 pont

Password

A következő beszélgetés egy feledékeny (Aladár) és egy nagyon okos (Balambér) matematikus között zajlott. Sajnos a beszélgetés rögzítése során néhány szó elveszett. Ami megmaradt:

A: Elfelejtettem a password-öm.

B: Mire emlékszel belőle?

A: A password-öm egy 9 jegyű szám. Az első három jegyére emlékszem: 189. Az is biztos, hogy 1-től 9-ig mindegyik számjegy szerepelt benne.

B: Ez még kevés információ.

A: A számnak van prímszám szomszédja. [Prímszám az a pozitív egész szám, aminek pontosan két osztója van.]

B: Sajnos még mindig nem tudok segíteni. Nem jut eszedbe még valami?

A: Az utolsó három számjegy szorzata: [itt hiányos a felvétel].

B: Sajnos még mindig nem lehet tudni... Hátha valami még eszedbe jut!

A: Az utolsó három számjegy összege: [itt hiányos a felvétel].

B: Megvan. Felírom neked egy papírra, hogy be tudj lépni a rendszerbe.

A: Köszönöm.

A fenti beszélgetés alapján határozd meg Aladár passwordjét!

A megoldás:
189265734

Magyarázat a megoldáshoz

1) Megkeressük azokat a számokat, amelyekben az összes számjegy pontosan egyszer szerepel 1-9 között, és az első 3 számjegy 189. Az összes lehetséges kombináció száma 6!, vagyis 720 lehetőség.

2) A 720 lehetőség közül ki kell válogatnunk azokat, amelyiknek valamelyik szomszédja prímszám. Minden számnak vesszük az alsó és felső szomszédját (tehát összesen 1440 számot), amelyekről meg kell vizsgálnunk, hogy prímek-e. Erre a legtöbb nyelvben van már valamilyen beépített vagy importálható megoldás. Eredményként 77 prímet kapunk, vagyis ennek megfelelően 77-re szűkítettük a lehetséges számokat (minden prímnek a megfelelő szomszédja, amiből eredetileg kiindultunk).

3) Ezután meg kell vizsgálnunk, hogy ezeknek a számoknak az utolsó 3 számjegye milyen összeget és szorzatot ad ki. Természetesen azoknál, ahol az utolsó 3 számjegy ugyanaz, ezek meg fognak egyezni, például:

189235746 (17, 168)

189352746 (17, 168)

Mindkét szám egyik szomszédja (189235747 és 189352747) prímszám, és az utolsó 3 számjegy összege és szorzata is ugyanaz. Olyan számot keresünk, ahol a számjegyek szorzatának ismeretében még nem egyértelmű a jelszó, ám a számjegyek összegének ismeretében egyértelmű lesz. (Ha a szorzat ismeretében már egyértelmű lenne, akkor Balambér nem kérdezett volna vissza. Ha pedig az összeg ismeretében továbbra sem lenne egyértelmű, akkor Balambér visszakérdezett volna.)
Először a lehetséges 77 szám közül válasszuk ki azokat, amelyekre igaz, hogy különböző 3 számjegy szorzata ugyanazt az eredményt adja. Ilyenek a 2,5,6 és 3,4,5 végűek (mert: 2*5*6 = 3*4*5 = 60), illetve ilyenek a 2,6,7 és 3,4,7 végűek (2*6*7 = 3*4*7 = 84). Ilyen módon a lehetséges jelszavak számát 11-re szűkíthetjük: 189265734, 189345672, 189345762, 189347526, 189354762, 189437256, 189437562, 189473652, 189734526, 189734562, 189743526.
Másodszor a lehetséges 11 szám közül válasszuk ki azokat, amelyekre igaz, hogy az utolsó 3 számjegy összege nem egyezik egyetlen másik szám összegével sem. Ilyen szám pedig csak egyetlenegy van.

189265734 (14, 84)

(Önmagában csak az összegből, vagy csak a szorzatból még nem lenne eldönthető, mert mindkettőre van olyan másik szám, ahol megegyezne.)

2. feladat 0 / 15 pont

Bit algoritmus

Adott az alábbi kód:

```
//A tömbök indexelése 0-tól indul
FUNCTION f(IN ARRAY OF INTEGERS a)
BEGIN
  r := 0;
  FOR i in 0..LENGTH(a)-1 DO
    r := r XOR a[i];
  END FOR;
  RETURN R;
END;
```

Jelölje **n** a tömb hosszát. Az alábbi állítások közül melyek igazak? Több helyes válasz lehetséges!

- [x] A fenti algoritmus megkeresi a tömb azon elemét, ami páratlan sokszor fordul elő a tömbben, feltéve hogy az összes többi elem páros sokszor fordul elő
- [x] A fenti algoritmus a tömb összes elemét össze-XOR-olja, és visszaadja az eredményt
- [x] A fenti algoritmus eredményében azok a bitek lesznek 1-esre állítva, amelyekre igaz, hogy a bejövő tömbben páratlan sokszor fordultak elő
- [ ] A fenti algoritmus átlagos futásideje: $\Theta(\log_2(n))$
- [x] A fenti algoritmus átlagos futásideje: $\Theta(n)$
- [ ] A fenti algoritmus átlagos futásideje: $\Theta(n*\log_2(n))$
- [ ] A fenti algoritmus átlagos futásideje: $\Theta(n^2)$
- [ ] A fenti algoritmus átlagos futásideje: $\Theta(n^3)$
- [ ] A fenti algoritmusban a bejövő egészek sorrendje releváns
- [x] A fenti algoritmusban a bejövő egészek sorrendje nem releváns

Magyarázat a megoldáshoz

3. feladat 0 / 15 pont

SQL injection

A következő sorokat látjuk valamilyen programkódban:

```
...
sql = "SELECT * FROM User WHERE Name ='\"" + IName + "\"";
...
EXEC sql INTO ResultSet;
...
```

Ha az "IName" változó értéke közvetlen a UI-ról jön, akkor mely szöveg esetén történhet sikeres SQL Injection? Egy SQL Injection akkor számít sikeresnek, ha segítségével extra információhoz juthat a az SQL Injection végrehajtója.

- ( ) joe OR "
- ( ) joe OR ""="
- (•) joe" OR ""="
- ( ) joe" OR "
- ( ) joe AND "
- ( ) joe AND ""="
- ( ) joe" AND ""="
- ( ) joe" AND "

Magyarázat a megoldáshoz

Vissza a kategóriáimhoz

Nyelvfüggetlen programozás és adatbáziskezelés (SAP)
7. forduló

Ismertető a feladathoz

Minden héten három izgalmas feladattal várunk.

Tekintettel arra, hogy egy választ sem rögzítettél az alábbi feladatlapon, ebben a fordulóban a kitöltésére rendelkezésre álló idő teljes egésze, azaz 60 perc került rögzítésre mint megoldáshoz felhasznált idő.

1. feladat 0 / 20 pont

Dominó

Adott egy **4xN**-es téglalap, aminek a bal felső és jobb alsó 1x1-es sarkát kivágtuk. Hányféleképpen lehet lefedni ezt a csonkított téglalapot 2x1-es dominókkal?

N=2 esetén a megoldás 0.

N=3 esetén a megoldás 5.

N=4 esetén a megoldás 0.

N=11 esetén a megoldás 2009.

Add meg a megoldás utolsó 9 számjegyét **N**=51-re!

A megoldás:
882998937

Magyarázat a megoldáshoz

A feladatot dinamikus programozással célszerű megoldani. A tábla 4 sorból és N oszlopból áll. Egy részfeladat 4 egész számmal írható le, és annak felel meg, hogy az egyes sorokban hány mezőt fedtünk már le dominókkal, a tábla bal oldalától kezdődően. A (1, 0, 0, 0) részfeladatot kell megoldani (a bal felső mezőt lefedettnek tekintjük). A végállás, azaz a (N, N, N, N-1) részfeladat megoldása 1 (0 dominó kell, egyféleképpen lehet 0 dominót lerakni), ha pedig valamelyik index ennél nagyobb, akkor az a részfeladat invalid, és 0 a megoldása. A rekurzió a következőképpen megy: tekintsük az (a, b, c, d) részfeladatot. Vegyük a fedetlen mezők közül a(z egyik) legbaloldalit. Az ezt a mezőt lefedő domínó maximum 3-féleképpen helyezkedhet el. a, b, c és d értékétől függ, hogy melyek lehetségesek ezek közül a helyzetek közül. A maradék lefedetlen halmaz mindhárom esetben egy valid részfeladatnak felel meg (ezért választottuk ki így a mezőt). Így a részfeladat megoldása maximum 3 másiknak az összege. A túlcsordulás elkerülése érdekében vagy tetszőleges pontossággal (Python), vagy modulo 1000000000 kell számolni. Ha szerencsénk van, belefér 64 bitbe a megoldás, de ez nagyobb N-re már nem garantált.

2. feladat 0 / 15 pont

Párhuzamos algoritmus

Adott az alábbi párhuzamosított kód. Vajon mi lehetett a program írójának a szándéka?

```
INPUT:
INTEGER n;
ARRAY OF INTEGERS v[0..n-1];
//n is power of 2, v[] is extended with zeros, if necessary

SHARED: ARRAY OF INTEGERS w[0..n-1];
INITIALIZE n THREADS IDENTIFIED FROM 0 TO n-1
LET EACH THREAD DO
  INTEGER id := GET_SELF_IDENTIFIER();
  w[id] := v[id];
INTEGER delta := n/2;
  WHILE delta > 0 DO
    SYNCHRONIZE ALL THREADS;
    IF id < delta THEN w[id] := w[id] + w[id+delta];
    delta := delta / 2;
  ENDDO;
ENDDO;

OUTPUT: w[0]
```

Mi a fenti algoritmus outputja, ha az inputja az 50000 alatti négyzetmentes(*) pozitív egészek csökkenő tömbje?

(*): Egy pozitív egész szám négyzetmentes, ha nem osztható 1-nél nagyobb négyzetszámmal. Például a 10 négyzetmentes, de a 18 nem négyzetmentes, mert osztható 9-cel.

A megoldás:
760089509

Magyarázat a megoldáshoz

Pl. CUDA kódban találkozunk gyakran ezzel a log N idejű aggregáló algoritmussal. Az eredmény itt konkrétan a tömb elemeinek összege. Az 1 és 50000 közötti négyzetmentes számokat kell tehát összeadni.

3. feladat 0 / 15 pont

Ismeretlen select

Adott az alábbi SELECT-mondat.

```
--A T tábla "a" nevű mezőjének a típusa INTEGER
SELECT DISTINCT A
FROM T AS T0
WHERE NOT EXISTS (
  SELECT 0
  FROM T AS T1 INNER JOIN T AS T2 ON T1.A>T2.A
  WHERE T0.A=T2.A
);
```

Mi a SELECT mondat eredménye, ha a T tábla "a" oszlopa tartalmazza az összes 1000000000-nál kisebb prímszámot(*)?

(*) Prímszám az a pozitív egész szám, amelynek pontosan két osztója van.

A megoldás:
999999937

Magyarázat a megoldáshoz

A lekérdezés azokat az elemeket adja vissza, amelyeknél nem létezik nagyobb elem ugyanabban a táblában. Vagyis a maximumot. Az 1000000000 alatti legnagyobb prímszám volt tehát a megoldás.

Vissza a kategóriáimhoz