1. **Vajon mit csinál? (5p)**

Adott az alg(x, b) algoritmus, amelynek bemeneti paraméterei az ***x*** és ***b*** természetes számok (1 ≤ ***x*** ≤ 1000, 1 < ***b*** ≤ 10). Határozzátok meg az algoritmus hatását.

|  |
| --- |
| **Algoritmus** alg(x, b):  s ← 0  **Amíg** x > 0 **végezd el**  s ← s + x **MOD** b  x ← x **DIV** b  **vége(amíg)**  **térítsd** s **MOD** (b - 1) = 0  **Vége(algoritmus)** |

kiszámítja az ***x*** természetes szám számjegyeinek összegét a ***b*** számrendszer­ben

vizsgálja, hogy az ***x*** szám számjegyeinek összege a ***b*** – 1 számrendszerben osztható-e (***b*** – 1)-gyel

vizsgálja, hogy az ***x*** szám osztható-e (***b*** – 1)-gyel

vizsgálja, hogy a ***b*** számrendszerben felírt ***x*** szám számjegyeinek összege osztható-e (***b*** – 1)-gyel

vizsgálja, hogy az ***x*** szám számjegyeinek összege osztható-e (***b*** – 1)-gyel

1. **Mit fog kiírni? (5p)**

Legyen a következő program:

|  |  |
| --- | --- |
| **C++/C** | |
| **int** sum(**int** n, **int** a[], **int** s){  s = 0;  **int** i = 1;  **while** (i <= n){  **if**(a[i] != 0)  s += a[i];  ++i;  }  **return** s;  } | **int** main(){  **int** n = 3, p = 0, a[10];  a[1] = -1;  a[2] = 0;  a[3] = 3;  **int** s = sum(n, a, p);  cout << s << ”;“ << p;  // printf("%d;%d", s, p);  **return** 0;  } |
| **Pascal** | |
| **type** vektor = **array**[1..10] **of** integer;  **function** sum(n: integer; a: vektor;  s: integer): integer;  **var** i : integer;  **begin**  s := 0;  i := 1;  **while** i <= n **do** **begin**  **if** a[i] <> 0 **then**  s := s + a[i];  i := i + 1;  **end**;  sum := s;  **end**; | **var** n, p, s: integer;  a: vektor;  **begin**  n := 3;  a[1] := -1;  a[2] := 0;  a[3] := 3;  p := 0;  s := sum(n, a, p);  writeln(s, ';', p);  **end**. |

Mit fog kiírni a program?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A.** 0;0 | **B.** 2;0 | **C.** 2;2 | **D.** Egyik válasz sem helyes | **E.** 0;2 |

1. **Logikai kifejezés (5p)**

Legyen a következő logikai kifejezés: (***X*** OR ***Z***) AND (NOT ***X*** OR ***Y***). Válasszátok ki ***X***, ***Y***, ***Z*** értékeit úgy, hogy a kifejezés értéke legyen *igaz*:

1. ***X*** = *hamis*; ***Y*** = *hamis*; ***Z*** = *igaz*;
2. ***X*** = *igaz*; ***Y*** = *hamis*; ***Z*** = *hamis*;
3. ***X*** = *hamis*; ***Y*** = *igaz*; ***Z*** = *hamis*;
4. ***X*** = *igaz*; ***Y*** = *igaz*; ***Z*** = *igaz*;
5. ***X*** = *hamis*; ***Y*** = *hamis*; ***Z*** = *hamis*;

1. **Számolás (5p)**

Legyen a számol(a, b) algoritmus, amelynek bemeneti paraméterei az ***a*** és ***b*** pozitív természetes számok, ahol 1 ≤ ***a*** ≤ 100, 1 ≤ ***b*** ≤ 100.

|  |  |
| --- | --- |
| 1.  2.  3.  4.  5.  6. | **Algoritmus** számol(a, b):  **Ha** a ≠ 0 **akkor**  **térítsd** számol(a **DIV** 2, b + b) + b \* (a **MOD** 2)  **vége(ha)**  **térítsd** 0  **Vége(algoritmus)** |

Az alábbi válaszok közül melyek hamisak?

1. ha ***a*** és ***b*** egyenlők, az algoritmus ***a*** értékét téríti
2. ha ***a*** = 1000 és ***b*** = 2, az algoritmus 10-szer hívja meg önmagát
3. az algoritmus által kiszámított és térített érték egyenlő (***a*** / 2 + 2 \* ***b***)-vel
4. az 5. sorban található utasítás egyszer sem hajtódik végre
5. az 5. sorban található utasítás egyszer hajtódik végre
6. **Elem beazonosítása(5p)**

Legyen az (1, 2, 3, 2, 5, 2, 3, 7, 2, 4, 3, 2, 5, 11, ...) sorozat, amelyet a követ­kezőképpen hoztunk létre: kiindulva a természetes számok sorozatából, azokat a számokat, amelyek nem prímszámok helyettesítettük a saját osztóikkal úgy, hogy minden ***d*** osztót csak egyszer használtunk minden szám esetében. Az alábbi algoritmusok közül melyik határozza meg a sorozat ***n***-dik elemét (***n*** természetes szám, 1 ≤ ***n*** ≤ 1000)?

|  |  |
| --- | --- |
| **A.** | **Algoritmus** beazonosítás(n):  a ← 1; b ← 1; c **←** 1  **Amíg** c < n **végezd el**  a **←** a + 1  b **←** a  c **←** c + 1  d **←** 2  **Amíg** c ≤ n **és** d ≤ a **DIV** 2 **végezd el**  **Ha** a **MOD** d **=** 0 **akkor**  c **←** c + 1  b **←** d  **vége(ha)**  d **←** d + 1  **vége(amíg)**  **vége(amíg)**  **térítsd** b  **Vége(algoritmus)** |

|  |  |
| --- | --- |
| **B.** | **Algoritmus** beazonosítás(n):  a **←** 1; b **←** 1; c **←** 1  **Amíg** c < n **végezd el**  c **←** c + 1; d **←** 2  **Amíg** c ≤ n **és** d ≤ a **DIV** 2 **végezd el**  **Ha** a **MOD** d **=** 0 **akkor**  c **←** c + 1; b **←** d  **vége(ha)**  d **←** d + 1  **vége(amíg)**  a **←** a + 1; b **←** a  **vége(amíg)**  **térítsd** b  **Vége(algoritmus)** |
| **C.** | **Algoritmus** beazonosítás(n):  a **←** 1; b **←** 1; c **←** 1  **Amíg** c < n **végezd el**  a **←** a + 1; d **←** 2  **Amíg** c < n **és** d ≤ a **végezd el**  **Ha** a **MOD** d **=** 0 **akkor**  c **←** c + 1; b **←** d  **vége(ha)**  d **←** d + 1  **vége(amíg)**  **vége(amíg)**  **térítsd** b  **Vége(algoritmus)** |

|  |  |
| --- | --- |
| **D.** | **Algoritmus** beazonosítás(n):  a **←** 1; b **←** 1; c **←** 1  **Amíg** c < n **végezd el**  b **←** a  a **←** a + 1  c **←** c + 1  d **←** 2  **Amíg** c ≤ n **és** d ≤ a **DIV** 2 **végezd el**  **Ha** a **MOD** d = 0 **akkor**  c **←** c + 1; b **←** d  **vége(ha)**  d **←** d + 1  **vége(amíg)**  **vége(amíg)**  **térítsd** b  **Vége(algoritmus)** |
| **E.** | **Algoritmus** beazonosítás(n):  a **←** 1; b **←** 1; c **←** 1  **Amíg** c < n **végezd el**  a **←** a + 1  b **←** a  c **←** c + 1  d **←** 2  f **←** *hamis*  **Amíg** c ≤ n **és** d ≤ a **DIV** 2 **végezd el**  **Ha** a **MOD** d = 0 **akkor**  c **←** c + 1; b **←** d  f **←** *igaz*  **vége(ha)**  d **←** d + 1  **vége(amíg)**  **Ha** f **akkor**  c **←** c - 1  **vége(ha)**  **vége(amíg)**  **térítsd** b  **Vége(algoritmus)** |

1. **Törzstényezők (5p)**

Legyen a törzsTényezők(n, d, k, x) algoritmus, amely meghatározza az ***n*** természetes szám ***k*** darab törzstényezőjét, a törzstényezők keresését a ***d*** értéktől kezdve. Bemeneti paraméterek az ***n***, ***d*** és ***k*** számok, kimeneti paraméterek az ***x*** sorozat, amely a ***k*** darab törzstényezőt tartalmazza (1 ≤ ***n*** ≤ 10000, 2 ≤ ***d*** ≤ 10000, 0 ≤ ***k*** ≤ 10000).

|  |
| --- |
| **Algoritmus** törzsTényezők(n, d, k, x):  **Ha** n **MOD** d = 0 **akkor**  k ← k + 1  x[k] ← d  **vége(ha)**  **Amíg** n **MOD** d = 0 **végezd el**  n ← n **DIV** d  **vége(amíg)**  **Ha** n > 1 **akkor**  törzsTényezők(n, d + 1, k, x)  **vége(ha)**  **Vége(algoritmus)** |

Határozzátok meg, hányszor hívja meg önmagát a törzsTényezők(n, d, k, x) algoritmus a következő programrészlet végrehajtásának következtében:

|  |
| --- |
| n ← 120  d ← 2  k ← 0  törzsTényezők(n, d, k, x) |

|  |  |  |  |
| --- | --- | --- | --- |
| **A.** 3-szor | **B.** 5-ször | **C.** 9-szer | **D.** 6-szor |

**E.** ugyanannyiszor, mint a következő programrészlet esetében:

|  |
| --- |
| n ← 750  d ← 2  k ← 0  törzsTényezők(n, d, k, x) |

**B rész (60 pont)**

1. **Konverzió (10 pont)**

Legyen a konverzió(s, hossz) algoritmus, amely átalakítja az ***s*** karakterlán­cot, amely egy 16-os számrendszerben ábrázolt szám, a megfelelő 10-es szám­rendszerben érvényes alakjára. Az ***s*** karakter­lánc ***hossz*** darab karaktert tartalmaz, ahol a karakter értéke lehet egy '0', '1', '2', '3', '4', '5', '6', '7', '8', '9' számjegy, vagy egy 'A', 'B', 'C', 'D', 'E', 'F' nagybetű (a ***hossz*** természetes szám, 1 ≤ ***hossz*** ≤ 10).

Írjátok le a konverzió(s, hossz) algoritmus *rekurzív* változatát úgy, hogy a fejléce és a hatása legyen azonos az alábbi algoritmuséval:

|  |
| --- |
| **Algoritmus** konverzió(s, hossz):  szám ← 0  **Minden** i = 1, hossz **végezd el**  **Ha** s[i]≥ 'A' **akkor**  szám ← szám \* 16 + s[i] - 'A' + 10  **különben**  szám ← szám \* 16 + s[i] - '0'  **vége(ha)**  **vége(minden)**  **térítsd** szám  **Vége(algoritmus)** |

1. **Azonos számjegyek (20 pont)**

Adott két természetes szám: ***a*** és ***b***, 1 ≤ ***a*** ≤ 1 000 000 és 1 ≤ ***b*** ≤ 1 000 000.

Írjatok algoritmust, amely meghatározza a ***k*** elemű ***x*** sorozatot, (***k*** – természe­tes szám, 0 ≤ ***k*** ≤ 1000), amely minden olyan természetes számot tárol, amelyek az [***a***, ***b***] intervallumhoz tartoznak és azonos számjegyekből állnak. Ha ilyen szám nem létezik, ***k*** értéke 0 lesz. Az algoritmus bemeneti paraméterei ***a*** és ***b***, kimeneti paraméterek pedig ***k*** és ***x***.

**1. *Példa***: ha ***a*** = 8 és ***b*** = 120, akkor ***k*** = 12 és ***x*** = (8, 9, 11, 22, 33, 44, 55, 66, 77, 88, 99, 111).

**2. *Példa***: ha ***a*** = 590 és ***b*** = 623, akkor ***k*** = 0 és az ***x*** sorozat üres.

1. **Sétáló robot (30 pont)**

Egy robot egy négyzetes tömb alakú térképen mozog, amelynek a mérete pá­ratlan számú, és a térkép minden celláján letesz bizonyos számú tárgyat. A robot a következő szabályok alapján mozog:

* a cellában, ahonnan indul letesz egy tárgyat, a második cellában, ahova lép, letesz kettőt, a harmadik cellában, ahova eljut, hármat és így tovább;
* a robot az utolsó oszlop közepéről indul, és lép egyet átlósan a felfele és jobbra szomszédos szabad cellára (párhuzamosan a mellékátlóval) ha ez a cella léte­zik és szabad a hely; ha ilyen cella nincs, akkor:
  + ha a robot az utolsó oszlopban található, akkor „átugrik” az első oszlopba, a robot felett levő sorba, ha ez a hely szabad;
  + ha a robot az első sorban található, akkor „átugrik” az utolsó sorba a robot­tól jobbra levő oszlopba, ha ez a hely szabad;
  + ha a robot a térkép jobb-felső sarkában található. akkor átugrik az utolsó sor első oszlopában levő cellába, ha ez a hely szabad.
* ha a cella, ahova lépni szeretne foglalt, a robot balra lép egyet a sorban, ahol éppen található, a szomszédos szabad cellába.

Ezek a szabályok biztosítják, hogy a robot a térkép minden celláját egyszer látogatja meg (és azt is, hogy nem fog elakadni sehol). Miután a robot a térkép minden cellájában elhelyezte a tárgyakat, megáll.

Például, egy 5 × 5 cellából álló térképen a robot első 22 lépése a következő:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 9 | 3 | 22 | 16 | 15 |
| 2 | 21 | 20 | 14 | 8 |
|  | 19 | 13 | 7 | 1 |
| 18 | 12 | 6 | 5 |  |
| 11 | 10 | 4 |  | 17 |

Írjatok algoritmust, amely meghatározza azoknak a tárgyaknak a ***szám*** számát, amelyeket a robot a térkép főátlóján levő cellákba helyez. Az algoritmus beme­neti paramétere a térkép ***n*** mérete (***n*** – páratlan természetes szám, 3 ≤ ***n*** ≤ 100**)**, kimeneti paramétere a ***szám*** (***szám*** – természetes szám).

**1. *Példa***: ha ***n*** = 5, akkor ***szám*** = 65;

**2. *Példa***: ha ***n*** = 11, akkor ***szám*** = 671.