**BABEŞ-BOLYAI TUDOMÁNYEGYETEM**

**MATEMATIKA-INFORMATIKA KAR**

**Matek-Info verseny – 1. Tételsor**

**Informatika írásbeli**

**A versenyzők figyelmébe:**

1. A tömböket 1-től kezdődően indexeljük.
2. A rácstesztekre (A rész) egy vagy több helyes válasz lehetséges. A válaszokat a vizsgadolgozatba írjátok (nem a feladatlapra). Ahhoz, hogy a feltüntetett pontszámot megkapjátok, elengedhetetlenül szükséges, hogy minden helyes választ megadjatok, és kizárólag csak ezeket.
3. A B részben szereplő feladatok megoldásait részletesen kidolgozva a vizsgadolgozatba írjátok.
   1. A feladatok megoldásait *pszeudokódban* vagy egy *programozási nyelvben* (*Pascal/C/C*++) kell megadnotok.
   2. A megoldások értékelésekor az első szempont az algoritmus ***helyessége***, majd a ***hatékonysága***, ami a *végrehajtási időt* és a *felhasznált memória méretét* illeti.
   3. A tulajdonképpeni megoldások előtt, ***kötelezően leírjátok szavakkal az alprogramokat, és megindokoljátok a megoldásotok lépéseit***. Feltétlenül írjatok ***megjegyzéseket*** (kommenteket), amelyek segítik az adott megoldás technikai részleteinek megértését. Adjátok meg az azonosítok jelentését és a fölhasznált adatszerkezeteket. Ha ez hiányzik, a tételre kapható pontszámotok 10%-kal csökken.
   4. Ne használjatok különleges fejállományokat, előredefiniált függvényeket (például STL, karakterláncokat feldolgozó sajátos függvények stb.).

**A rész (30 pont)**

1. **Vajon mit csinál? (5p)**

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

|  |
| --- |
| **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:

|  |  |  |
| --- | --- | --- |
| **Varianta C++/C** |  | **Varianta Pascal** |
| 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;  } |  | type vector = array[1..10] of integer;  function sum(n:integer; a:vector; 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 : vector;  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?

1. 0;0
2. 2;0
3. 2;2
4. Egyik válasz sem helyes
5. 0;2
6. **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 TRUE:

1. ***X*** ← FALSE; ***Y*** ← FALSE; ***Z*** ← TRUE;
2. ***X*** ← TRUE; ***Y*** ← FALSE; ***Z*** ← FALSE;
3. ***X*** ← FALSE; ***Y*** ← TRUE; ***Z*** ← FALSE;
4. ***X*** ← TRUE; ***Y*** ← TRUE; ***Z*** ← TRUE;
5. ***X*** ← FALSE; ***Y*** ← FALSE; ***Z*** ← FALSE;
6. **Számolás (5p)**

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

1 ≤ ***a*** ≤ 1000, 1 ≤ ***b*** ≤ 1000.

|  |  |
| --- | --- |
| 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 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)?

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 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)** |  | **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)** |
|  | **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)** |  | **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)** |
|  | **Algoritmus** beazonosítás(n):OKOKOK  a **←** 1, b **←** 1, c **←** 1  **Amíg** c < n **végezd el**  a **←** a + 1, b **←** a, c **←** c + 1, d **←** 2  f **←** false  **Amíg** c ≤ n **és** d ≤ a **DIV** 2 **végezd el**  **Ha** a **MOD** d = 0 **akkor**  c **←** c + 1, b **←** d, f **←** true  **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*** 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)

1. 3-szor
2. 5-ször
3. 9-szer
4. 6-szor
5. 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***, ahol 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 szám, 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.

**Megjegyzések:**

1. Minden tétel kidolgozása kötelező.
2. A piszkozatokat nem vesszük figyelembe.
3. Hivatalból jár 10 pont.
4. Rendelkezésetekre áll 3 óra.

**JAVÍTÁSI KULCS**

**HIVATALBÓL 10 pont**

**A rész 30 pont**

**A. 1.** c, d  **5 pont**

**A. 2.** b  **5 pont**

**A. 3.** a, d  **5 pont**

**A. 4.** a, c, d  **5 pont**

**A. 5.** e  **5 pont**

**A. 6.** a, e  **5 pont**

**B rész 60 pont**

**B. 1. Konverzió 10 pont**

– fejléc a követelményeknek megfelelően 2 pont

– a rekurzív hívások leállási feltétele 1 pont

– a leálláskor térített érték 1 pont

– nem számjegy karakter feltétele 2 pont

– térített érték, amikor a karakter nem számjegy 2 pont

– térített érték, amikor a karakter számjegy 2 pont

|  |
| --- |
| **Algoritmus** konverzió(s, hossz):  **Ha** hossz > 0 **akkor**  **Ha** s[hossz] ≥ 'A' **akkor**  **térítsd** konverzió(s, hossz - 1) \* 16 + s[hossz] - 'A' + 10  **különben**  **térítsd** konverzió(s, hossz - 1) \* 16 + s[hossz] - '0'  **vége(ha)**  **különben**  **térítsd** 0  **vége(ha)**  **Vége(algoritmus)** |

**B. 2. Azonos számjegyek 20 pont**

* V1: az azonos számjegyeket tartalmazó számok – **számolásokkal történő generálása** 20 pont

o bemeneti és kimeneti paraméterek a követelményeknek megfelelően 2 pont

o a számok generálása (egyszámjegyű számok és/vagy 11, 111, 1111, többszörösei...) 16 pont

o az azonos számjegyű számok sorozatba mentése 2 pont

* V2: az [***a***, ***b***] intervallum számainak **bejárása** és **vizsgálata** 10 pont

o bemeneti és kimeneti paraméterek a követelményeknek megfelelően 2 pont

o annak a tulajdonságnak az ellenőrzése, hogy a szám számjegyei azonosak 4 pont

o az [***a***, ***b***] intervallumhoz tartozó számok bejárása az ellenőrzés céljából 2 pont

o az azonos számjegyű számok sorozatba mentése 2 pont

**B. 3. Sétáló robot 30 pont**

* V1: a kért eredmény helyes meghatározása számolással: **(*n* \* *n* \* *n* + *n*) / 2** 30 pont
  + bemeneti és kimeneti paraméterek a követelményeknek megfelelően 2 pont
  + számolás 14 pont
  + a számolások részletes megindoklása 14 pont
* V2: a kért eredmény helyes meghatározása szimulálással 25 pont
  + bemeneti és kimeneti paraméterek a követelményeknek megfelelően 2 pont
  + az összes ***n ×* *n***cella bejárása 4 pont
  + helyes mozgás (a 4 lehetséges esetben) 4\*4 pont
  + a főátlón található tárgyak számának helyes meghatározása 3 pont

**B rész (megoldások) 60 pont**

**B. 1. Konverzió 10 pont**

|  |
| --- |
| **Algoritmus** konverzió(s, hossz):  **Ha** hossz > 0 **akkor**  **Ha** s[hossz] ≥ 'A' **akkor**  **térítsd** konverzió(s, hossz - 1) \* 16 + s[hossz] - 'A' + 10  **különben**  **térítsd** konverzió(s, hossz - 1) \* 16 + s[hossz] - '0'  **vége(ha)**  **különben**  **térítsd** 0  **vége(ha)**  **Vége(algoritmus)** |

**B. 2. Azonos számjegyek 20 pont**

* V1: az azonos számjegyeket tartalmazó számok – **számolásokkal történő generálása** 20 pont

|  |
| --- |
| **void** azonosSzamjegyek(**int** a, **int** b, **int** &k, **int** x[]){  k = 0;  **for**(**int** i = a; ((i < 10) && (i <= b)); i++){  x[++k] = i;  }  **int** akt = 11;  **int** leptek = 11;  **while**(akt <= b){  **if**(akt >= a)  x[++k] = akt;  akt = akt + leptek;  **if**(akt > 9 \* leptek){  akt = leptek \* 10 + 1;  leptek = akt;  }  }  } |

**B. 3. Sétáló robot 30 pont**

* V1: a kért eredmény helyes meghatározása számolással: **(*n* \* *n* \* *n* + *n*) / 2** 30 pont

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 9 | 3 |  |  |  |  | 17 | 23 |  |  |  |  |  |  | 22 | 16 | 15 |  | 9 | 3 | 22 | 16 | 15 |
| 2 |  |  |  | 8 |  | 24 |  |  |  | 18 |  |  | 21 | 20 | 14 |  |  | 2 | 21 | 20 | 14 | 8 |
|  |  | 13 | 7 | 1 |  |  |  | 13 | 19 | 25 |  | 25 | 19 | 13 |  |  |  | 25 | 19 | 13 | 7 | 1 |
|  | 12 | 6 | 5 |  |  |  | 14 | 20 | 21 |  |  | 18 |  |  |  | 24 |  | 18 | 12 | 6 | 5 |  |
| 11 | 10 | 4 |  |  |  | 15 | 16 | 22 |  |  |  |  |  |  | 23 | 17 |  | 11 | 10 | 4 |  | 17 |

***n*** = 5

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | 24 |  |  |
|  |  |  | 23 | **17** |  | 10 |
| 9 |  | 22 | 16 | 15 | **9** | 3 |
|  | **21** | 20 | 14 | 8 | 2 |  |
| 25 | 19 | **13** | 7 | 1 |  |  |
| 18 | 12 | 6 | **5** |  |  |  |
| 11 |  | 4 |  | 17 |  |  |

kialakul ***n*** átló, de minket a főátlón levő elemek érdekelnek:

* az első átlón az 1, 2, ..., ***n*** számok találhatók, ahonnan ***n*** bekerül az összegbe
* a 2. átlón az ***n*** + 1, ***n*** + 2, ..., 2***n*** számok találhatók, ahonnan az összegbe bekerül 2***n*** – 1
* a 3. átlón az 2***n*** + 1, 2***n*** + 2, ..., 3***n*** számok találhatók, ahonnan az összegbe bekerül 3***n*** – 2
* ...
* az ***n***. átlón az (***n*** – 1) \* ***n*** + 1, ..., ***n*** \* ***n*** számok találhatók, ahonnan az összegbe bekerül (***n*** – 1) \* ***n*** + 1 (vagyis ***n*** \* ***n*** – (***n*** – 1))

Így a főátlón található elemek összege:

***n*** + (2***n*** – 1) + (3***n*** – 2) + ... + ***n*** \* ***n*** – (***n*** – 1) =

***n*** + 2***n*** + 3***n*** + ... + ***n*** \* ***n*** – (1 + 2 + ... + (***n*** –1)) =

***n*** \* (1 + 2 + ... + ***n***) – (***n*** – 1) \* ***n*** / 2 = (az első ***n***, illetve ***n*** – 1 természetes szám összege)

***n*** \* ***n*** \* (***n*** + 1) /2 – (***n*** – 1) \* ***n*** / 2 = (kiemeljük az (1 / 2)-őt és elvégezzük a szorzásokat)

(***n*** \* ***n*** \* ***n*** + ***n*** \* ***n*** – ***n*** \* ***n*** + ***n***) / 2 = (***n*** \* ***n*** \* ***n*** + ***n***) / 2

Ha ***n***= 5, a térítendő összeg 65.

|  |
| --- |
| **int** targyak(**int** n){  **return** (n \* n \* n + n) / 2;  } |