**BABEŞ-BOLYAI TUDOMÁNYEGYETEM**

**MATEMATIKA-INFORMATIKA KAR**

**Felvételi verseny – 1. Tételsor**

**Informatika írásbeli**

**A versenyzők figyelmébe:**

1. Minden tömböt 1-től kezdődően indexelünk.
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 (algoritmusokat), 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 stb. 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 a kifejezés(n) algoritmus, ahol ***n*** egy természetes szám (1 ≤ ***n*** ≤ 10000).

|  |
| --- |
| **Algoritmus** kifejezés(n):  **Ha** n > 0 **akkor**  **Ha** n **MOD** 2 = 0 **akkor**  **térítsd** -n \* (n + 1) + kifejezés(n - 1)  **különben**  **térítsd** n \* (n + 1) + kifejezés(n - 1)  **vége(ha)**  **különben**  **térítsd** 0  **vége(ha)**  **Vége(algoritmus)** |

Állapítsátok meg az *E*(*n*) kifejezésnek azt a matematikai alakját, amelyet a fenti algoritmus számít ki:

* 1. E(*n*) = 1\* 2 - 2 \* 3 + 3 \* 4 + ... + (-1)*n*+1 \* *n* \* (*n* + 1)
  2. E(*n*) = 1\* 2 - 2 \* 3 + 3 \* 4 + ... + (-1)*n* \* *n* \* (*n* + 1)
  3. E(*n*) = 1\* 2 + 2 \* 3 + 3 \* 4 + ... + (-1)*n*+1 \* *n* \* (*n* + 1)
  4. E(*n*) = 1\* 2 + 2 \* 3 + 3 \* 4 + ... + (-1)*n* \* *n* \* (*n* + 1)
  5. E(*n*) = 1\* 2 - 2 \* 3 - 3 \* 4 - ... - (-1)*n* \* *n* \* (*n* + 1)

1. **Számolás (5p)**

Adott a számol(n) algoritmus, ahol ***n*** egy természetes szám (1 ≤ ***n*** ≤ 10000).

|  |
| --- |
| **Algoritmus** számol(n):  x ← 0, z ← 1  **Amíg** z ≤ n **végezd el**  x ← x + 1  z ← z + 2 \* x  z ← z + 1  **vége(amíg)**  **térítsd** x  **Vége(algoritmus)** |

Az alábbi válaszok közül melyek **hamisak**?

1. Ha ***n*** = 25 vagy ***n*** = 35, akkor a számol(n) 5-öt térít vissza
2. Ha ***n*** < 8, akkor a számol(n) 3-at térít vissza
3. Ha ***n*** ≥ 85 és ***n*** < 100, akkor a számol(n) 9-et térít vissza
4. Az algoritmus kiszámítja és visszatéríti az ***n***-nél kisebb, szigorúan pozitív négyzetszámok darabszámát
5. Az algoritmus kiszámítja és visszatéríti az ***n*** szám négyzetgyökének egész részét
6. **Logikai kifejezés (5p)**

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

1. ***X*** ← *hamis*; ***Y*** ← *hamis*; ***Z*** ← *hamis*;
2. ***X*** ← *hamis*; ***Y*** ← *hamis*; ***Z*** ← *igaz*;
3. ***X*** ← *hamis*; ***Y*** ← *igaz*; ***Z*** ← *hamis*;
4. ***X*** ← *igaz*; ***Y*** ← *hamis*; ***Z*** ← *igaz*;
5. ***X*** ← *hamis*; ***Y*** ← *igaz*; ***Z*** ← *igaz*;
6. **Mit fog kiírni? (5p)**

Legyen a következő program:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **C változat** | **C++ változat** | **Pascal változat** | | #**include** <stdio.h>  **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; **int** p = 0;  **int** a[10];  a[1] = -1; a[2] = 0; a[3] = 3;  **int** s = sum(n, a, &p);  printf("%d;%d", s, p);  **return** 0;  } | #**include** <iostream>  **using namespace** std;  **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; **int** p = 0;  **int** a[10];  a[1] = -1; a[2] = 0; a[3] = 3;  **int** s = sum(n, a, p);  cout << s << ";" << p;  **return** 0;  } | **type** vector = **array**[1..10] **of** integer;  **function** sum(n : integer; a : vector;  var 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);  write(s,';',p);  **end**. | |  |

A végrehajtás eredményeként mit fog kiírni a program?

1. 3;0
2. 2;0
3. 0;2
4. 2;2
5. Egyik válasz sem helyes
6. **Szerencsés szám (5p)**

Egy nullától különböző ***x*** természetes számot *szerencsés*nek nevezzük, ha a négyzete felírható ***x*** darab egymás utáni természetes szám összegeként. Például, a 7 szerencsés szám, mivel 72 = 4 + 5 + 6 + 7 + 8 + 9 + 10.

A következő algoritmusok közül, melyik dönti el az ***x*** természetes számról (2 ≤ ***x*** ≤ 1000), hogy *szerencsés szám*? Minden algoritmus bemeneti paramétere az ***x*** szám, kimeneti paraméterei pedig a nullától különböző ***start*** természetes szám és a ***szerencsés*** logikai változó. Ha az ***x*** szerencsés szám, akkor ***szerencsés*** = *igaz* és ***start*** értéke az összeg első tagjának értéke (például, ha ***x*** = 7, akkor ***start*** = 4); ha ***x*** nem szerencsés szám, akkor ***szerencsés*** = *hamis* és ***start*** értéke -1.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Algoritmus** szerencsésSzám(x, start, szerencsés):  xNégyzet ← x \* x  szerencsés ← *hamis*  start ← -1, k ← 1, s ← 0  **Amíg** k ≤ xNégyzet - x **és** **nem** szerencsés **végezd el**  **Minden** i ← k, k + x - 1 **végezd el**  s ← s + i  **vége(minden)**  **Ha** s = xNégyzet **akkor**  szerencsés ← *igaz*  start ← k  **vége(ha)**  **vége(amíg)**  **Vége(algoritmus)** |  | **Algoritmus** szerencsésSzám(x, start, szerencsés):  xNégyzet ← x \* x  szerencsés ← *hamis*  start ← -1, k ← 1  **Amíg** k ≤ xNégyzet - x **és** **nem** szerencsés **végezd el**  s ← 0  **Minden** i ← k, k + x - 1 **végezd el**  s ← s + i  **vége(minden)**  **Ha** s = xNégyzet **akkor**  szerencsés ← *igaz*  start ← k  **vége(ha)**  k ← k + 1  **vége(amíg)**  **Vége(algoritmus)** |
|  | **Algoritmus** szerencsésSzám(x, start, szerencsés):  **Ha** x **MOD** 2 = 0 **akkor**  szerencsés ← *hamis*  start ← -1  **különben**  szerencsés ← *igaz*  start ← (x + 1) **DIV** 2  **vége(ha)**  **Vége(algoritmus)** |  | **Algoritmus** szerencsésSzám(x, start, szerencsés):  **Ha** x **MOD** 2 = 0 **akkor**  szerencsés ← *hamis*  start ← -1  **különben**  szerencsés ← *igaz*  start ← x **DIV** 2  **vége(ha)**  **Vége(algoritmus)** |
|  | **Algoritmus** szerencsésSzám(x, start, szerencsés):  szerencsés ← *igaz*  start ← (x + 1) **DIV** 2  **Vége(algoritmus)** |  |  |

1. **Tegyél 'b' betűket (5p)**

Legyen az ***n*** × ***n*** méretű négyzetes ***mat*** tömb (***n*** – páratlan természetes szám, 3 ≤ ***n*** ≤ 100). A tegyélB(mat, n, i, j) algoritmus 'b' betűket tesz a ***mat*** tömb bizonyos pozícióira. Az ***i*** és ***j*** paraméterek természetes számok (1 ≤ ***i*** ≤ ***n***, 1 ≤ ***j*** ≤ ***n***).

|  |
| --- |
| **Algoritmus** tegyélB(mat, n, i, j):  **Ha** i ≤ n **DIV** 2 **akkor**  **Ha** j ≤ n - i **akkor**  mat[i][j] ← 'b'  tegyélB(mat, n, i, j + 1)  **különben**  tegyélB(mat, n, i + 1, i + 2)  **vége(ha)**  **vége(ha)**  **Vége(algoritmus)** |

Határozzátok meg, hányszor hívja meg önmagát a tegyélB(mat, n, i, j) algoritmus a következő programrészlet végre­hajtásának következtében:

|  |
| --- |
| n ← 7, i ← 2, j ← 4  tegyélB(mat, n, i, j) |

1. 5-ször
2. ugyanannyiszor, mint a következő programrészlet esetében:

n ← 9, i ← 3, j ← 5

tegyélB(mat, n, i, j)

1. 10-szer
2. 0-szor
3. végtelenszer

**B rész (60 pont)**

1. **Számolás - karakterekkel (10 pont)**

Legyen a számolásKarakterekkel(s, n, p, q, szám) algoritmus, ahol ***s*** egy ***n*** karakterből álló sorozat (***n*** természetes szám, 1 ≤ ***n*** ≤ 9), ***p***, ***q*** és ***szám*** természetes számok (1 ≤ ***p*** ≤ ***n***, 1 ≤ ***q*** ≤ ***n***, ***p*** ≤ ***q***).

|  |
| --- |
| **Algoritmus** számolásKarakterekkel(s, n, p, q, szám):  eredmény ← 0  i ← p  **Amíg** i ≤ q **végezd el**  **Amíg** i ≤ q **és** s[i] ≥ '0' **és** s[i] ≤ '9' **végezd el**  szám ← szám \* 10 + s[i] - '0'  i ← i + 1  **vége(amíg)**  eredmény ← eredmény + szám  szám ← 0  i ← i + 1  **vége(amíg)**  **térítsd** eredmény  **Vége(algoritmus)** |

Írjátok le a számolásKarakterekkel(s, n, p, q, szám) algoritmus *rekurzív* változatát úgy, hogy a fejléce és a hatása legyen azonos a fenti algoritmuséval. Az alábbi programrészletből hívjuk meg:

**Beolvas:** n, s, p, q

**KiÍr:** számolásKarakterekkel(s, n, p, q, 0)

1. **Periódus (25 pont)**

Azt mondjuk, hogy egy ***n*** karakterből álló sorozatnak a *periódusa* ***k***, ha az adott sorozat előállítható egy ***k*** elemű ka­raktersorozat ismételt egymás után ragasztása révén (2 ≤ ***n*** ≤ 200, 1 ≤ ***k*** ≤ 100, 2 \* ***k*** ≤ ***n***). Az "abcabcabcabc" sorozatnak a *periódusa* 3, mivel előállítható úgy, hogy az "abc" karaktersorozatot 4-szer egymás után ragasztjuk; ugyanakkor a sorozatnak a *periódusa* 6, ha úgy tekintjük, hogy az "abcabc" karaktersorozatot 2-szer egymás után ragasztjuk. Az "abcxabc" sorozatnak nincs periódusa. *Maximális periódus*nak nevezzük a sorozat legnagyobb *periódusát*.

Írjatok algoritmust, amely meghatározza az ***n*** elemű ***x*** karaktersorozat (***n*** – természetes szám, 2 ≤ ***n*** ≤ 100) ***pm*** *maximális periódusát*. Ha az ***x*** sorozatnak nincs periódusa, ***pm*** értéke -1. Az algoritmus bemeneti paraméterei ***x*** és ***n***, kimeneti paramétere ***pm***.

**1. *Példa:*** ha ***n*** = 8 és ***x*** = "abababab", akkor ***pm*** = 4.

**2. *Példa:*** ha ***n*** = 7 és ***x*** = "abcxabc", akkor ***pm*** = -1.

1. **Robi-kert (25 pont)**

Egy modern műszaki megoldásokat kedvelő kertész elhatározta, hogy a kert ágyásainak öntözéséhez egy robotokból álló „hadsereget” fog használni. A vizet a kertet átszelő fősétány végénél található forrásból szeretné venni. Minden ágyáshoz kioszt egy robotot úgy, hogy minden robotnak egyetlen ágyást kell megöntöznie. Minden robot egyszerre indul öntözni a forrástól reggel ugyanabban az időpontban (például, reggel 5:00:00 órakor) és párhuzamosan dolgoznak, meg­állás nélkül azonos időn át. A robotok haladnak a fősétányon, amíg a saját ágyásukhoz érnek, az ágyást megöntözik és visszatérnek a forráshoz, hogy a víztartályukat újból megtöltsék. A tevékenységre szánt idő lejárta után, minden robot egyszerre leáll, függetlenül attól, hogy mely állapotban vannak. Eredetileg, a forrásnál egyetlen vízcsap volt. A kertész észrevette, hogy öntözés közben késések adódtak, mivel a robotoknak sorba kellett állniuk a vízcsapnál, hogy feltölt­hessék a víztartályukat. Ebből kifolyólag úgy döntött, hogy fel fog szerelni több vízcsapot. A robotok reggel tele víz­tar­tállyal indulnak. Két robot nem töltheti fel a víztartályát ugyanabban a pillanatban egyazon vízcsapnál.

Ismert adatok: a ***t*** időintervallum (másodpercekben) amennyi ideig az ***n*** robot dolgozik, a ***di*** a másodpercek száma, amennyi idő alatt a robotok eljutnak a forrástól a számukra kiosztott ágyásig, az ***ui*** a másodpercek száma, amennyi idő alatt a robotok megöntözik a saját ágyásukat. Még ismert, hogy mindegyik robot a saját víztartályát egy másodperc alatt tölti meg. (***t***, ***n****,* ***di***, ***ui*** – természetes számok, 1 ≤ ***t*** ≤ 20000, 1 ≤ ***n*** ≤ 100, 1 ≤ ***di*** ≤ 1000, 1 ≤ ***ui*** ≤ 1000, ***i*** = 1, 2, ..., ***n***).

**Követelmények:**

1. Soroljátok fel azokat a robotokat, amelyek találkoznak a forrásnál egy adott ***mt*** időpillanatban (1 ≤ ***mt*** ≤ ***t***). Indo­koljátok meg a választ.

Megjegyzés: a robotokat a sorszámaik alapján azonosítjuk be.

1. Legkevesebb hány ***minÚjVízcsap*** új vízcsapot kell felszerelnie a kertésznek ahhoz, hogy a robotoknak egyáltalán ne kelljen várakozniuk egymás után, amikor meg kell tölteniük a víztartályukat? Indokoljátok meg a választ.
2. Írjatok algoritmust, amely meghatározza, hogy legkevesebb hány ***minÚjVízcsap*** újabb vízcsapot kell még fel­sze­relnie a kertésznek. Bemeneti paraméterek ***n***, ***t***, valamint a ***d*** és ***u*** sorozatok, mindkettő ***n*** elemmel, kimeneti paraméter a ***minÚjVízcsap***.

**1. *Példa:*** ha ***t*** = 32 és ***n*** = 5, ***d*** = (1, 2, 1, 2, 1), ***u*** = (1, 3, 2, 1, 3), akkor ***minÚjVízcsap*** = 3. Magyarázat: az első ágyással foglalkozó robotnak szüksége van egy másodpercre, hogy az ágyáshoz érjen, egy másodpercre az öntözéshez és még egy másodpercre ahhoz, hogy visszatérjen a forráshoz; így, ez a robot 1 + 1 + 1 = 3 másodperc után tér vissza a forráshoz, hogy újra megtöltse a tartályát az indulási időponttól számítva (5:00:00), tehát 5:00:03-kor; megtölti a víztartályát egy másodperc alatt és 5:00:04-kor indul vissza az ágyáshoz; visszatér 5:00:07-kor a forráshoz, és így tovább; tehát az első, a második, a negyedik és az ötödik robot találkoznak a forrásnál 05:00:23-kor; következik, hogy 3 új vízcsapra van szükség.

**2. *Példa:*** ha ***t*** = 37, ***n*** = 3, ***d*** = (1, 2, 1), ***u*** = (1, 3, 2), akkor ***minÚjVízcsapok*** = 1.

**Megjegyzések:**

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

**MEGOLDÁS**

**A rész**

**A. 1.** Vajon mit csinál? Válasz: A

**A. 2.** Számol. Válaszok: B, D

**A. 3.** Logikai kifejezés. Válaszok: A, B, D, E

**A. 4.** Mit ír ki? Válasz: D

**A. 5**. Szerencsés szám. Válaszok: B, C

**A. 6.** Tegyél 'b' betűket. Válaszok: A, B

**B rész**

**B. 1. Számolás karakterekkel 10 pont**

* bemeneti paraméterek 2 pont
* leállási feltétel 1 pont
* visszatérített érték a rekurzióból való kilépéskor 1 pont
* feltétel, amely ellenőrzi, hogy a karakter nem számjegy 2 pont
* visszatérített érték, ha a karakter *nem* számjegy 2 pont
* visszatérített érték, ha a karakter számjegy 2 pont

**Algoritmus** számolásKarakterekkelRekurzívan(s, n, p, q, sz):

**Ha** p > q **akkor**

**térítsd** sz

**különben**

**Ha** s[p] < '0' **vagy** s[p] > '9' **akkor**

**térítsd** sz + számolásKarakterekkelRekurzívan(s, n, p + 1, q, 0)

**különben**

**térítsd** számolásKarakterekkelRekurzívan(s, n, p + 1, q, 10 \* sz + s[p] - '0')

**vége(ha)**

**vége(ha)**

**Vége(algoritmus)**

**B. 2. Periódus 25 pont**

* bemeneti paraméterek 2 pont
* a periódus lehetséges értékeinek bejárása legtöbb 10 pont

Megjegyzés:

* a pontszám függ attól, hogy csak a lehetséges értékeket veszi számításba
* a periódus hossza ***n*** osztója
* a periódus ellenőrzése legtöbb 13 pont

Megjegyzés: a pontszám függ a felhasznált adatszerkezetek számától

**Algoritmus** ellenőriz(n, x, periódus):

**Minden** i = periódus, n - 1 **végezd el:**

**Ha** x[i + 1] ≠ x[i **MOD** periódus + 1] **akkor**

**térítsd** *hamis*

**vége(ha)**

**vége(minden)**

**térítsd** *igaz*

**Vége(algoritmus)**

**Algoritmus** maxPeriódus(n, x):

periódus ← -1

**Minden** per = 2, per \* per ≤ n **végezd el:** { *lehetséges periódus hosszúságok: 2..int(sqrt(n)),* }

{ *de a per <= sqrt(n) helyett per \* per <= n* }

**Ha** n **MOD** per = 0 **akkor** { *ahhoz, hogy "per" periódus legyen, osztania kell n-t* }

**Ha** ellenőriz(n, x, n **DIV** per) **akkor** { *ha n / per valóban periódus* }

**térítsd** n **DIV** per

**vége(ha)**

**Ha** per \* per < n **és** ellenőriz(n, x, per) **akkor** { *ha n / per nem volt jó, a következő, amit ellenőrzünk per* }

periódus ← per { *ha per megfelelt, ő a periódus* }

**vége(ha)**

**vége(ha)** { *folytatódik a Minden, de van egy „tartalék” periódusunk* }

**vége(minden)**

**térítsd** periódus

**Vége(algoritmus)**

**B. 3. Robi-kert.................................................................................................................................................. 25 pont**

Ha ***n*** = 5, ***d*** = (1, 2, 1, 2, 1), ***u*** = (1, 3, 2, 1, 3), ***t*** = 32, kiszámítjuk ***q*** értékeit, amely értékek az egyes robot időigényeit tárolják az ágyásokhoz, és vissza a vízcsaphoz:

2 \* 1 + 1 + 1 = 4,

2\* 2 + 3 + 1 = 8,

2 \* 1 + 2 + 1 = 5,

2 \* 2 + 1 + 1 = 6,

2 \* 1 + 3 + 1 = 6

=> (4, 8, 5, 6, 6)

Inicializáljuk a ***t*** = 32 elemű ***v*** vektort 0 értékekkel. Vesszük rendre a ***q*** értékeit, és minden robot esetében növeljük a ***q*** többszöröseinek megfelelő elemeket 1-gyel. Ezek az értékek a megfelelő időpillanatban szükséges vízcsapok számát fejezik ki.

***q*** = 4

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | **4** | 5 | 6 | 7 | **8** | 9 | 10 | 11 | **12** | 13 | 14 | 15 | **16** | 17 | 18 | 19 | **20** |
| ***v*** | 0 | 0 | 0 | **1** | 0 | 0 | 0 | **1** | 0 | 0 | 0 | **1** | 0 | 0 | 0 | **1** | 0 | 0 | 0 | **1** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 21 | 22 | 23 | **24** | 25 | 26 | 27 | **28** | 29 | 30 | 31 | **32** |
| ***v*** | 0 | 0 | 0 | **1** | 0 | 0 | 0 | **1** | 0 | 0 | 0 | **1** |

***q*** = 8

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| ***v*** | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 1 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
| ***v*** | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 2 |

***q*** = 5

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| ***v*** | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 2 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 2 | 0 | 0 | 0 | 2 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
| ***v*** | 0 | 0 | 0 | 2 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 2 |

***q*** = 6

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| ***v*** | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 2 | 0 | 1 | 0 | 2 | 0 | 0 | 1 | 2 | 0 | 1 | 0 | 2 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
| ***v*** | 0 | 0 | 0 | 3 | 1 | 0 | 0 | 1 | 0 | 2 | 0 | 2 |

***q*** = 6

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| ***v*** | 0 | 0 | 0 | 1 | 1 | 2 | 0 | 2 | 0 | 1 | 0 | 3 | 0 | 0 | 1 | 2 | 0 | 2 | 0 | 2 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
| ***v*** | 0 | 0 | 0 | 4 | 1 | 0 | 0 | 1 | 0 | 3 | 0 | 2 |

Meghatározzuk a ***v*** tömb maximumát. A példa esetében a maximum 4 (a 24. időpillanatban), tehát szükséges összesen 4 vízcsap, vagyis a létező egyen kívül még 4 – 1 = 3 új vízcsap.

**Válaszok a kérdésekre**

1. Soroljátok fel azokat a robotokat, amelyek találkoznak a forrásnál egy adott ***mt*** időpillanatban (1 ≤ ***mt*** ≤ ***t***). Indo­koljátok meg a választ. Megjegyzés: a robotokat a sorszámaik alapján azonosítjuk be.

**Válasz:** Egy adott ***mt*** (1 ≤ ***mt*** ≤ ***t***) időpillanatban azok a robotok találkoznak a forrásnál, amelyeknek esetében ***q*** értéke az ***mt*** többszöröse (egyenlő az ágyáshoz vezető út megtételéhez, a vissza út megtételéhez, az öntözéshez és a tartály megtöltéséhez szükséges idővel) 3 pont

1. Legkevesebb hány ***minÚjVízcsap*** új vízcsapot kell felszerelnie a kertésznek ahhoz, hogy a robotoknak egyáltalán ne kelljen várakozniuk egymás után, amikor meg kell tölteniük a víztartályukat? Indokoljátok meg a választ.

**Válasz:** A legkevesebb vízcsap, amit a kertésznek fel kell szerelnieegyenlőa ***v*** tömb maximumával, amiből kivonunk egyet (a már létező vízcsapot), ahol ***v*** elemei azoknak a robotoknak a darabszámát tárolják, amelyek az egyes időpillanatokban találkoznak a forrásnál 2 pont

1. Az algoritmus:

* Egy gyakoriság-tömb használata minden robot munkaidejének többszöröseire vonatkozóan 10 pont

c.1. a bemeneti és kimeneti paraméterek tiszteletben tartása **2 pont**

c.2. munkaidő kiszámítása (***q*** = 2 \* útidő + öntözés + tartály feltöltése) **1 pont**

c.3. a gyakoriság-tömb feldolgozása **4 pont**

c.3.1. inicializálás 1 pont

c.3.2. aktualizálás 3 pont

* + - 1. v1a vagy v1b: bejárás ***q***-ról ***q***-ra 3 pont
      2. v2a sau v2b: bejárás egyesével és annak vizsgálata, hogy ***q*** többszöröse vagy sem 2 pont

c.4. a maximális vízcsapszám meghatározása 2 **pont**

* + - 1. párhuzamosan az aktualizálással vagy az aktualizálások után 1 pont

c.5. a felszerelendő új vízcsapok számának meghatározása (max - 1) **1 pont**

**Algoritmus** robiKert(n, d, u, t):

max ← 1

**Minden** i = 1, t **végezd el:**

aux[i] ← 0 { aux[i]-*ban tároljuk az* i. *pillanatban szükséges vízcsapokat* }

**vége(minden)**

**Minden** j = 1, n **végezd el:**

q ← d[j] \* 2 + u[j] + 1

**vége(minden)**

**Minden** i = q, t, q végezd el: { *növeljük a* q *többszöröseit az* aux *tömbben* }

aux[i] ← aux[i] + 1

**Ha** max < aux[i] **akkor** { *meghatározzuk az* aux *tömb maximumát* }

max ← aux[i]

**vége(ha)**

**vége(minden)**

**térítsd** max - 1

**Vége(algoritmus)**

* Szimulálás 10 pont