1. **Mit ír ki? (6 pont)**

Adott az alábbi program. Állapítsátok meg, mit ír ki a program a végrehajtás eredményeként.

|  |  |
| --- | --- |
| 1. P(a, b) = 253;   a = 11; b = 11; | 1. P(a, b) = 132;   a = 11; b = 121; |
| 1. P(a, b) = 22;   a = 11; b = 11; | 1. P(a, b) = 253;   a = 11; b = 121; |

|  |  |
| --- | --- |
| **C++** | **C** |
| **#include<iostream>**  **using namespacestd;**  **int** P(**int** x, **int** &y){  y = y \* x;  x = x + y;  **return** x + y;  }  **int** main(){  **int** a = 11; **int** b = a;  cout << "P(a, b) = " << P(a, b);  cout << endl << "; a = " << a;  cout << "; b = " << b << endl;  **return** 0;  } | **#include<stdio.h>**  **int** P(**int** x, **int** \*y){  \*y = (\*y) \* x;  x = x + (\*y);  **return** x + (\*y);  }  **int** main(){  **int** a = 11; **int** b = a;  printf("P(a, b) = %d", P(a, &b));  printf(";\na = %d", a);  printf("; b = %d\n", b);  **return** 0;  } |
| **Pascal** | |
| **function** P(x : Integer; **var** y : Integer) : Integer;  **begin**  y := y \* x;  x := x + y;  P := x + y;  **end**;  **var** a, b : Integer;  **Begin**  a := 11; b := a;  Write('P(a, b) = ', P(a, b))  WriteLn('; ');  Write('a = ', a);  WriteLn('; b = ', b);  **End**. | |

1. **Kiértékelés (6 pont)**

Adott az ***n*** elemű (1 ≤ ***n*** ≤ 10 000), 30 000-nél kisebb természetes számokat tároló ***v*** sorozat. Állapít­sátok meg, hogy a következő programrészletek közül me­lyiket kellene a „…” helyére írni, ahhoz, hogy a feldolgoz(v, n, er, m) algo­ritmus ismétlő struktúrájának végrehajtása után az ***er*** sorozat, a ***v*** sorozat azon elemeit tárolja, amelyek 5-nek azon többszörösei, amelyek páros indexű helyeken találhatók. Az ***er*** sorozat hosszát az ***m*** változó tárolja.

|  |  |
| --- | --- |
| **Algoritmus** feldolgoz(v, n, er, m)  m ← 0  **Minden** i = 1, n **végezd el**  ...  **vége(minden)**  **Vége(algoritmus)** | |
| **A.** | **Ha** i **MOD** 2 = 0 **és** v[i] **MOD** 5 = 0 **akkor**  m ← m + 1  er[m] ← v[i]  **vége(ha)** | **B.** | **Ha** i **MOD** 2 = 0 **akkor**  **Ha** v[i] **MOD** 10 = 5 **akkor**  m ← m + 1  er[m] ← v[i]  **vége(ha)**  **Ha** v[i] **MOD** 10 = 0 **akkor**  m ← m + 1;  er[m] ← v[i]  **vége(ha)**  **vége(ha)** |
| **C.** | **Ha** v[i] **MOD** **10** = 0 **akkor**  **Ha** i **MOD** 2 = 0 **akkor**  m ← m + 1  er[m] ← v[i]  **különben**  **Ha** v[i] **MOD** 10 = 5 **akkor**  m ← m + 1;  er[m] ← v[i]  **vége(ha)**  **vége(ha)**  **vége(ha)** | **D.** | **Ha** i **MOD** 2 = 0 **és** v[i] **MOD** 10 = 5  **és** v[i] **MOD** 10 = 0 **akkor**  m ← m + 1  er[m] ← v[i]  **vége(ha)** |

1. **Vajon mit csinál? (6 pont)**

Legyen a találdKi(n) algoritmus, ahol ***n*** természetes szám (1 ≤ ***n*** ≤ 10000). Állapítsátok meg a találdKi(n) algoritmus hatását.

|  |  |
| --- | --- |
| **Algoritmus** találdKi(n)  f ← 0  p ← -1  **Minden** c = 0, 9 **végezd el**  x ← n  k ← 0  **Amíg** x > 0 **végezd el**  **Ha** x **MOD** 10 = c **akkor**  k ← k + 1  **vége(ha)**  x ← x **DIV** 10  **vége(amíg)**  **Ha** k > f **akkor**  p ← c  f ← k  **vége(ha)**  **vége(minden)**  **térítsd** p  **Vége(algoritmus)** | 1. Kiszámítja és visszatéríti a 10-es számrend­szerben megadott ***n*** szám számjegyeinek da­rabszámát. 2. Kiszámítja és visszatéríti a 10-es számrend­szerben megadott ***n*** szám legnagyobb szám­jegyének darabszámát. 3. Kiszámítja és visszatéríti a 10-es számrend­szerben megadott ***n*** szám egyik olyan szám­jegyét, amelynek előfordulási száma maxi­mális. 4. Kiszámítja és visszatéríti a 10-es számrend­szerben megadott ***n*** szám, ***c***-vel egyenlő számjegyeinek darabszámát. |

1. **Sajátos sorozat generálása (6 pont)**

Ismert, hogy egy ***k*** elemű ***x*** sorozat (***x*** = (***x***1, ***x***2, ***x***3, ..., ***xk***)) ***p*** hosszúságú tömb­szakasza az ***x*** sorozat ***p*** darab eleméből áll, amelyek egymás utáni helyeket fog­lalnak el. Például, ***y*** = (***x***3, ***x***4, ***x***5, ***x***6) egy ***p*** = 4 hosszúságú tömbszakasz.

Legyen az ***M*** ={1, 2, ..., ***n***} számjegyeket tároló halmaz, és az ***M*** halmaz ***n***! darab permutációja (***n*** – természetes szám, ***n*** ≤ 9). Létre lehet hozni azt a legrö­videbb ***ss*** számjegysorozatot, amely ***M*** számjegyeit tárolja és az ***M*** halmaz összes ***n*!** permutációja megtalálható az ***ss***-ben, ***n*** hosszúságú tömbszakaszok formájá­ban.

Például, ha ***n*** = 3, a permutációk száma 6, az ***ss*** sorozatot 9 számjegy alkotja és lehetne, például ***ss*** = (1, 2, 3, 1, 2, 1, 3, 2, 1). A ***perm*1** = (1, 2, 3) permutációnak felhasználjuk az utolsó két számjegyét és hozzáfűzünk egy harmadik számjegyet, ahhoz, hogy egy másik permutációt kapjunk, vagyis a ***perm*2** = (2, 3, 1) permutá­ciót; majd a ***perm*2** két utolsó számjegye után helyezhetjük a 2-es számjegyet és megkapjuk a ***perm*3** = (3, 1, 2) permutációt. Ha az (1, 2) után írnánk a 3-as szám­jegyet, megkapnánk a ***perm*1** permutációt, amely már létezik az eddig generált számjegysorozatban. Ekkor ***perm*3**-nak csak az utolsó számjegyét használjuk fel és egy olyan permutációt keresünk, amely a 2-es számjeggyel kezdődik és még nincs benne a sorozatban stb. Így, a létrehozott sorozatban egyetlen olyan három számjegyből álló tömbszakasz található, amely nem permutáció: (1, 2, 1); a többi tömbszakasz, (vagyis (2, 1, 3), (1, 3, 2) és (3, 2, 1)) helyesek.

Állapítsátok meg, hogy az ***M*** = {1, 2, 3, 4} halmaz esetében legkevesebb hány számjegy típusú eleme lesz az ***ss*** sorozatnak.

**A.** 55 **B.** 16 **C.** 33 **D.** 37

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | ***perm*1** | | |  | | | | | | | 1 | 2 | 3 | 1 | 2 | 1 | 3 | 2 | 1 | |  | ***perm*2** | | |  | | | | | |

1. **Sorozat feldolgozása (6 pont)**

Adott a feldolgoz(n, x) algoritmus, ahol ***x*** egy ***n* –** 1 elemű, különböző ter­mészetes számokat tároló sorozat, amely számok az {1, 2, …, ***n***} halmazhoz tar­toznak. Állapítsátok meg az algoritmus által visszatérített érték jelentését.

|  |
| --- |
| **Algoritmus** feldolgoz(n, x)  s ← 0  **Minden** i = 1, n - 1 **végezd el**  s ←s + x[i]  **vége(minden)**  **térítsd** n \* (n + 1) / 2 - s  **Vége(algoritmus)** |

1. Az algoritmus az első ***n*** nem nulla természetes szám összegének és az ***x*** soro­zat elemei összegének különbségét téríti vissza.
2. Az algoritmus az első ***n*** nem nulla természetes szám összegének és az ***x*** soro­zat elemei összegének (kivéve az utolsó elemet) különbségét téríti vissza.
3. Az algoritmus annak az {1,2, …, ***n***} halmazhoz tartozó természetes számnak az értékét téríti, amely nem szerepel az ***x*** sorozatban.
4. Az algoritmus 0-t térít vissza.
5. **Egészítsétek ki (6 pont)**

Adott az ***n*** elemű (3 ≤ ***n*** ≤ 100), növekvően rendezett ***x*** sorozat, amely 30 000-nél kisebb különböző természetes számokat tartalmaz. A legközelebbi(x, bal, jobb, ér) algoritmus meghatározza az ***x*** sorozat legnagyobb értékű elemének pozícióját, amely a ***bal*** és ***jobb*** pozíciók között helyezkedik el (1 ≤ ***bal*** < ***jobb*** ≤ ***n***) és, amelynek az értéke kisebb, vagy egyenlő ***ér***-rel. Ha nem létezik ilyen elem, a legközelebbi(x, bal, jobb, ér) algoritmus 0-t térít vissza.

A modulusz(a) algoritmus az ***a*** egész szám abszolút-értékét téríti vissza.

A számol(n, x, adottSz) algoritmus meghatározza azt az elemét az ***x*** soro­zatnak, amely a legköze­lebb áll ***adottSz***-hoz. Ha két elem azonosan közel van ***adottSz*** értékéhez, az algoritmus a nagyobb szá­mot határozza meg.

Legyen ***n*** = 5, ***x*** = (5, 9, 11, 15, 99) és ***adottSz*** = 12. Állapítsátok meg melyik kifejezéssel helyette­síthető a „…” a legközelebbi(x, bal, jobb, ér) algoritmus­ban, ahhoz, hogy a számol(n, x, adottSz) algoritmus 11-et térítsen vissza.

|  |
| --- |
| **Algoritmus** legközelebbi(x, bal, jobb, ér)  **Ha** ér > x[jobb] **akkor térítsd** jobb  **vége(ha)**  **Ha** ér < x[bal] **akkor** **térítsd** bal - 1  **vége(ha)**  közép ← (bal + jobb) **DIV** 2  **Ha** ... **akkor**  **térítsd** közép - 1  **különben**  **Ha** ér < x[közép] **akkor**  **térítsd** legközelebbi(x, bal, közép - 1, ér)  **különben**  **térítsd** legközelebbi(x, közép + 1, jobb, ér)  **vége(ha)**  **vége(ha)**  **Vége(algoritmus)** |
| **Algoritmus** számol(n, x, adottSz)  i ← legközelebbi(x, 1, n, adottSz)  **Ha** i = 0 **akkor**  **térítsd** x[i + 1]  **különben**  **Ha** modulusz(x[i]- adottSz) < modulusz(x[i + 1] - adottSz) **akkor**  **térítsd** x[i]  **különben**  **térítsd** x[i + 1]  **vége(ha)**  **vége(ha)**  **Vége(algoritmus)** |

1. x[közép - 1] <= ér **és** ér < x[közép]
2. x[közép - 1] <= ér **vagy** ér < x[közép]
3. x[közép - 1] < ér **és** ér <= x[közép]
4. x[közép] <= ér **és** ér < x[közép - 1]
5. **Lovagok és hazugok (6 pont)**

Egy szigeten csak lovagok élnek, akik mindig igazat mondanak és hazugok, akik mindig hazudnak. Egy látogató, aki a szigetre érkezett szeretné eldönteni két helybéli lakos természetét, akikkel találkozna a szigeten. Így, amikor találkozik két lakossal, A-val és B-vel, felteszi a ***Q*1** kérdést A-nak: „Mindketten lovagok vagytok?” de a kapott válasz (***V*1**) alapján nem tudja eldönteni, hogy milyen ter­mészetű a két lakos. Ezért, a látogató feltesz egy újabb ***Q*2** kérdést A-nak: „Egyformák vagytok, mindketten lovagok vagy mindketten hazugok?”; ezúttal, a kapott válasz (***V*2**) alapján a látogató már megvilágosodik (vagyis, most már tudja, hogy melyik lakos lovag és melyik hazug).

Az alábbi változatok közül, melyik felel meg az A lakos válaszainak, tudva azt, hogy a látogató pontosan meg tudta állapítani a két helybéli lakos természetét.

1. ***V*1**: Igen, ***V*2**: Igen
2. ***V*1**: Igen, ***V*2**: Nem
3. ***V*1**: Nem
4. A **B.** és **C.** válaszok helyesek
5. **Számolás – karakterekkel**

Legyen a számolásKarakterekkel(s, n, i, szám) algoritmus, ahol ***s*** egy ***n*** karakterből álló sorozat (***n*** természetes szám, 1 ≤ ***n*** ≤ 200), ***i*** és ***szám*** természetes számok (1 ≤ ***i*** ≤ ***n***).

|  |
| --- |
| **Algoritmus** számolásKarakterekkel(s, n, i, szám):  eredmény ← 0  **Amíg** i ≤ n **végezd el**  **Amíg** i ≤ n **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, i, szám) algoritmus *rekurzív* vál­tozatát úgy, hogy a fejléce és a hatása legyen azonos a fenti algoritmuséval. Az alábbi programrészletből hívjuk meg:

|  |
| --- |
| **Beolvas:** n, s  **KiÍr:** számolásKarakterekkel(s, n, 1, 0) |

1. **Iteratívból rekurzív**

Legyen a következő alprogram, ahol az ***a*** és ***b*** bemeneti paraméterek termé­szetes számok (0 < ***a*** ≤ 1 000 és 0 < ***b*** ≤ 1 000):

|  |  |
| --- | --- |
| **Algoritmus** f(a, b):  eredmény ← 0  **Amíg** a > 0 **végezd el:**  **Ha** a **MOD** 2 = 1 **akkor**  eredmény ← eredmény + b  **vége(ha)**  a ← a **DIV** 2  b ← b + b  **vége(amíg)**  **térít** eredmény  **Vége(algoritmus)** | * 1. Adjátok meg annak a feladatnak a szö­vegét, amelyet ez az algoritmus old meg.   2. Mit térít az f(5, 15) hívás?   3. Írjátok le az adott algoritmus *rekurzív* változatát, amelynek fejléce azonos az iteratív (nem rekurzív) algoritmus fejlé­cével. |

**B. Banánok**

Egy hajótörés után a tengerészeknek sikerült a mentőcsóna­kokkal megmenekülni. Minden csónakban volt három tenge­rész és minden csónak más-más szigeten vetődött partra. Éle­lem után kellett nézniük és minden ***i***. szigeten a tengerészek összegyűjtöttek ***bi*** banánt, amelyeket elraktározták a csónakjuk­ba, mivel úgy döntöttek, hogy csak másnap osztják szét egymás között. Bármely csónakban legtöbb ***k*** banán fér el. Az éjszaka folyamán, az egyik szigeten, az egyik tengerész felkelt és szétosztotta a csónakban levő banánokat három részre, minden halomba ugyanannyi banánt tett, de maradt még egy banán, amit nem tehetett egyik halomba sem, így ezt megette. Ezután, az egyik részt elrejtette, a másik két részt visszatette a csónakba és lefeküdt aludni. Reggelig, minden ten­gerész, mindegyik szigeten elvégezte ugyanezt a titkos beavatkozást (elosztotta a banánokat három egyenlő részre és megette az egyetlen megmaradt banánt). Reggel, mindegyik szigeten, a megmaradt banánokat három egyenlő (nem üres) részre osztották és megint maradt egy banán, amit a tengerészek egy majomnak adtak. Igen, mindegyik szigeten élt egy-egy majom!☺

**Követelmények**:

1. Ha az egyik szigeten, az éjszaka beállta előtt, a tengerészek összegyűjtöttek 241 banánt, hány banán maradt egy-egy halomban az osztozkodás végén (ezen a szigeten)? (2 pont)
2. Ha az egyik szigeten, az osztozkodás végén minden halom 15 banánt tar­talmazott, és egy másik szigeten minden halom 31 banánt, hány banánt gyűjtöttek összesen a két szigeten? (2 pont)
3. Írjatok alprogramot, amely egy adott ***k*** számra kiszámítja azt a lehetséges legnagyobb ***bmax*** értéket, amely azoknak a banánoknak a száma, amelye­ket a tengerészek egy bizonyos szigeten összegyűjtöttek. Bemeneti para­méter: ***k***. Kimeneti paraméter: ***bmax***. Keressetek képletet, amely kiszámítja ***bmax*** legnagyobb értékét ***k*** függvényében. Magyarázzátok el az alkalma­zott gondolat­menetet. (***bmax***, ***k*** – természetes számok, 1 ≤ ***bmax*** ≤ ***k*,** 100 ≤ ***k*** ≤ 10 000 000). (14 pont)

***Példa:*** ha ***k*** = 200, akkor ***bmax*** = 160.

1. Írjatok alprogramot, amely egy adott ***k*** számra kiszámítja az összes szigeten összegyűjtött maxi­mális banánszámok összegét (***összegMax***). Tudjuk, hogy minden szigeten a tengerészek különbö­ző ***bi*** számú banánt gyűjtöttek (***bi*** – természetes szám, ***i*** = 1, 2, ..., ***sz*,** 1 ≤ ***bi*** ≤ ***k*,** 100 ≤ ***k*** ≤ 10 000 000). Bemeneti paraméterek: ***k*** és ***sz*** – a szigetek száma (***sz*** – természetes szám, 2 ≤ ***sz*** ≤ 10). Kimeneti paraméter: ***összegMax*** – az összes szigeten össze­gyűjtött maximális banánszámok összege. A ***k*** és ***sz*** változók értékei úgy vannak megadva, hogy a feladatnak legyen megoldása. (12 pont)

***Példa:*** ha ***k*** = 400, ***sz*** = 3, akkor a három szigeten rendre 322, 241 és 160 banánt gyűjtöttek. Következik, hogy az összes szigeten összegyűjtött ma­ximális banánszámok összege ***összegMax =*** 322 + 241 + 160 = 723.