Concurs de admitere
Facultatea de Matematică și Informatică
Universitatea de Vest din Timișoara

Sesiunea Iulie, 2023

---

1. Timpul total de lucru: 3 ore
2. Toate subiectele sunt obligatorii!
3. Citiți cu atenție informațiile legate de tipurile de întrebări, prezentate mai jos.

Tipuri de întrebări

Pe foaia de concurs veți întâlni trei tipuri de întrebări. Pentru a le recunoaște mai ușor, în partea dreaptă veți regăsi un șablon pentru răspunsuri, cu indicații legate de tipul de întrebare.

Tipul 1 Pentru întrebările de acest tip veți marca pe foaia de concurs un singur răspuns corect. Le veți recunoaște după modelul indicat în partea dreaptă a acestui rând.

MODEL, marcați exact un răspuns:
(1) (2) (3) (4)

Tipul 2 Pentru întrebările de acest tip veți marca pe foaia de concurs fie un singur răspuns corect, fie două răspunsuri corecte. Le veți recunoaște după modelul indicat în partea dreaptă a acestui rând.

MODEL, marcați 1-2 răspunsuri:
(1) (2) (3) (4) (5)

Tipul 3 Pentru întrebările de acest tip veți marca pe foaia de concurs fie un singur răspuns corect, fie două răspunsuri corecte, fie trei răspunsuri corecte. Le veți recunoaște după modelul indicat în partea dreaptă a acestui rând.

MODEL, marcați 1-3 răspunsuri:
(1) (2) (3) (4) (5)

Punctaj

Fiecare variantă corectă de răspuns valorează exact 2 puncte.
Dacă răspunsul oferit identifică toate variantele corecte de răspuns, la punctajul întrebării se adaugă un bonus de 1 punct pentru răspuns complet.
Numărul maxim de puncte care poate fi obținut este 98. La acest punctaj se adaugă 10 puncte din oficiu.

Problema 1: Un cod simplu

Dacă `PERICOL` devine `CIPROLE`, și `ROZALIU` devine `LARZIUO` atunci `TROPICE` devine `.....`

Ciornă, marcați exact un răspuns:
(1) (2) (3) (4)

(1) `RICOPET`
(2) `IPTOCER` ✔
(3) `PORECIT`
(4) `OEPRTIC`

Explicații

Regula de codificare este: 1->3, 2->7, 3->4, 4->2, 5->1, 6->5, 7->6. (Prima literă ajunge poziția a treia, a doua pe poziția a șaptea, etc).
Se aplică regula de codificare de mai sus și se obține: 1->3 , 2->7 , 3->4 , 4->2 , 5->1 , 6->5 , 7->6 ==> IPTOCER.

Problema 2: O secvență

Se consideră următoarea secvența de numere:

$$1, 50, 3, 46, 7, 36, 13, 16, x, y$$

Identificați valorile x și y de mai sus, apoi oferiți răspunsul corect.

Ciornă, marcați exact un răspuns:
(1) (2) (3) (4)

(1) $x = 25, y = 6$
(2) $x = 21, y = -18$ ✔
(3) $x = 27, y = -6$
(4) $x = 23, y = 18$

Explicații

Regula pentru pozițiile impare: 3=1+2, 7=3+4, 13=7+6, urmează 21=13+8.
Regula pentru pozițiile pare: 46=50-(1+3), 36=46-(3+7), urmează 16-(21+13)=-18

Problema 3: Selfies

Cinci persoane sunt aliniate pentru a fi fotografiate împreună. Stângu stă la stânga lui Right, dar la dreapta lui Brilhante. Milieu stă la dreapta lui Right, iar Estremo este între Right și Milieu.

Care dintre următoarele afirmații este corectă:

Ciornă, marcați exact un răspuns:
(1) (2) (3) (4)

(1) Milieu stă la dreapta lui Estremo. ✔
(2) Brilhante este al doilea din dreapta.
(3) Stângu este primul din stânga.
(4) Right este penultimul din listă.

Explicații

Alinierea este: Brilhante–Stângu–Right–Estremo–Milieu

Problema 4: Cuvinte

Un copil de 2 ani se joacă cu literele `A`, `B`, `C`, `R`, având 5 din fiecare. După 2 ore de joacă ajunge la cuvântul `BARCA`. Câte cuvinte distincte de 5 litere poate forma în total? Dar folosind doar literele cuvântului `BARCA`?

Ciornă, marcați exact un răspuns:
(1) (2) (3) (4)

(1) 3125, 60
(2) 1024, 120
(3) 3125, 1024
(4) 1024, 60 ✔

Explicații

Numărul de cuvinte de 5 litere care se pot construi folosind literele din mulțimea $\{A, B, C, R\}$ coincide cu numărul de funcții definite pe mulțimea cu 5 elemente (pozițiile din cuvânt) care iau valori dintr-o mulțime cu 4 elemente (literele din mulțime), adică este $4^5 = 1024$.
Numărul total de cuvinte care se pot construi folosind 5 litere distincte este $5! = 120$. Dintre literele cuvantului `BARCA`, două sunt identice (A), deci numărul de cuvinte distincte este $120/2 = 60$

Problema 5: Cărți de joc

MisteryX are în mână un pachet de cărți de joc, pe care-l folosește pentru următorul joc: întoarce o carte de joc; dacă este o carte de joc roșie, elimină un număr de cărți de joc cel mult egal cu numărul de cărți roșii întoarse până în acel moment. Jocul se termină atunci când nu mai are cărți de joc în mână.

Care dintre următoarele bucăți de algoritm ar putea modela acest joc?

Ciornă, marcați exact un răspuns:
(1) (2) (3) (4)

① ✔

```
n ← 0
cât timp există cărți în pachet execută
    întoarce cartea următoare
    dacă este o carte roșie atunci
        n ← n+1
        pentru n times execută
            elimină o carte
```

②

```
cât timp există cărți în pachet execută
    întoarce cartea următoare
    cât timp cartea curentă este roșie execută
        elimină o carte
```

③

```
n ← 0
cât timp există cărți în pachet execută
    întoarce cartea următoare
    pentru n times execută
        elimină o carte
        dacă cartea a fost roșie atunci
            n ← n+1
```

④

```
n ← 0
cât timp există cărți în pachet execută
    întoarce cartea următoare
    cât timp cartea curentă este roșie execută
        n ← n+1
        elimină o carte
        întoarce cartea următoare
```

Explicații

Varianta (2) elimină o singură carte la întalnirea unei cărți roșii. Varianta (3) elimină carți și la întalnirea unei cărți care nu este roșie, iar varianta (4) elimină o singură carte pentru fiecare carte roșie întoarsă. Așadar doar prima variantă ar putea modela jocul.

Problema 6: Pseudocod 1-2

Se consideră doi algoritmi scriși în pseudocod, în care toate variabilele sunt numere întregi, iar $\mathbf{m} \geq 1$.

```
procedura VAR 1
    suma ← 0
    pentru i ← 1 to m execută
        suma ← suma + i
    întoarce suma
```

```
suma ← 0
i ← ⟨valoare_initiala⟩
cât timp ⟨conditie⟩ execută
    i ← i + 1
    suma ← suma + i
întoarce suma
```

procedura VAR 2

Care din următoarele expresii pot înlocui $\langle valoare\_initiala \rangle$ și $\langle conditie \rangle$, astfel încât VAR 2 să producă același rezultat cu VAR I?

Ciornă, marcați 1-2 răspunsuri: ① ② ③ ④ ⑤

① $i \leftarrow 0$ și $i \leq m - 1$ ✔

② $i \leftarrow 0$ și $i < m + 1$

③ $i \leftarrow 1$ și $i \leq m$

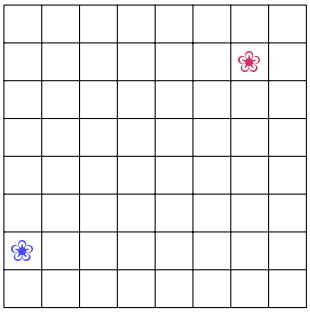
④ $i \leftarrow 1$ și $i < m - 1$

⑤ $i \leftarrow 0$ și $i < m$ ✔

Explicații

Algoritmul `Var 1` calculează suma primelor m numere naturale. Întrucât în algoritmul `Var 2` incrementarea lui i este înainte de actualizarea sumei, este necesar ca i să fie inițializată cu 0 (răspunsuri 1, 2 și 5); răspunsul 2 nu este corect pentru că se adună și $m + 1$ la sumă.

Problema 7: Doar sus sau dreapta



O problemă relativ simplă: estimați numărul total de drumuri, folosind doar direcțiile SUS și DREAPTA, prin care se poate ajunge de la floarea albastră la floarea mov.

① mai puțin de 400

② exact 256

③ exact 462 ✔

④ peste 400 ✔

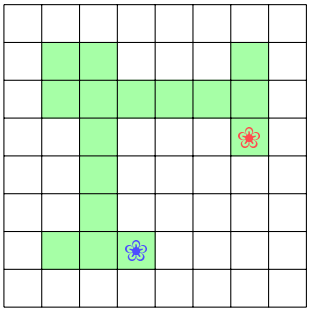
⑤ exact 924

Ciornă, marcați 1-2 răspunsuri:
① ② ③ ④ ⑤

Explicații

Oricare dintre trasee necesită 11 pași dintre care 6 pot fi efectuați pe orizontală și 5 pe verticală (în orice combinație). Numărul de trasee este prin urmare $C_{11}^5 = C_{11}^6 = 462$.

Problema 8: Doar verde



În această problemă vom ajuta un roboțel care se mișcă pe suprafața din imagine, parcurgând toate pătratele verzi, să realizeze o deplasare de la floarea albastră la floarea roșie.

Se presupune că `pozInițial` și `pozFinal` sunt deja inițializate, funcția PozUrm() returnează poziția următoare, în direcția de înaintare, funcția Avans() avansează pe următoarea poziție, în direcția de înaintare, DirStânga() schimbă direcția de înaintare cu 90 de grade, la stânga.

Folosind aceste notații, care dintre următorii algoritmi i-ar putea permite roboțelului să se deplaseze între cele două flori, acesta fiind amplasat inițial pe poziția florii albastre, într-o direcție de înaintare oarecare.

①
```
poziție = pozInițial
repetă
    dacă PozUrm() este VERDE atunci
        Avans()
    altfel
        DirStânga()
până când poziție = pozFinal
```

② ✔
```
poziție = pozInițial
repetă
    dacă PozUrm() este VERDE  ȘI
        PozUrm() <> pozInițial atunci
        Avans()
    altfel
        DirStânga()
până când poziție = pozFinal
```

③
```
poziție = pozInițial
repetă
    dacă PozUrm() este ALB atunci
        DirStânga()
    altfel
        Avans()
până când poziție = pozFinal
```

④ ✔
```
poziție = pozInițial
repetă
    dacă PozUrm() este ALB  SAU
        PozUrm() = pozInițial atunci
        DirStânga()
    altfel
        Avans()
până când poziție = pozFinal
```

Ciornă, marcați 1-2 răspunsuri:
① ② ③ ④ ⑤

⑤ Niciunul dintre algoritmii de mai sus

Explicații

Variantele corecte (2 și 4) sunt singurele care evită revenirea în poziția inițială (ceea ce ar duce la un ciclu infinit, deci va patrula permanent).

Problema 9: Cu tact

O platformă hardware folosește câteva instrucțiuni de bază pentru execuția programelor, fiecare necesitând un anumit număr de cicluri de tact la execuție, conform specificațiilor de mai jos:

1. <*instrucțiune*$_N$> — folosește N cicluri de tact (e.g. <*instrucțiune*$_1$> folosește 1 ciclu de tact)
2. <*condiție*$_M$> — folosește M cicluri de tact (e.g. <*condiție*$_2$> folosește 2 cicluri de tact)
3. IF <*condiție*$_M$> THEN <*instrucțiune*$_N$> — folosește $4+N+M$ cicluri de tact
4. WHILE$_K$ <*condiție*$_M$> DO <*instrucțiune*$_N$> — folosește minim $8+M$ și maxim $K \times (4+M+N)$ cicluri de tact, unde K este numărul de iterații
5. NOT <*condiție*$_M$> — folosește $1+M$ cicluri de tact
6. <*condiție*$_{M1}$> AND <*condiție*$_{M2}$> — folosește minim $2+M1$ și maxim $2+M1+M2$ cicluri de tact
7. <*condiție*$_{M1}$> OR <*condiție*$_{M2}$> — folosește folosește minim $2+M1$ și maxim $2+M1+M2$ cicluri de tact

Fiind dată următoarea secvență de cod:

```
WHILE8 (<condiție2> AND (NOT <condiție4>)) DO
      IF (<condiție2> OR <condiție2>) THEN
            <instrucțiune4>
```

Precizați care dintre următoarele variante sunt corecte:

Ciornă, marcați 1-2 răspunsuri:
(1) (2) (3) (4) (5)

(1) Secvența folosește întotdeauna peste 200 de cicluri de tact.

(2) Secvența folosește uneori mai mult de 200 de cicluri de tact. ✔

(3) Secvența folosește cel mult 200 de cicluri de tact.

(4) Secvența folosește uneori mai puțin de 12 cicluri de tact.

(5) Secvența nu folosește mai puțin de 12 cicluri de tact. ✔

Explicații

Numărul maxim de cicluri de tact: $8 \cdot (4 + cond(2+2+(1+4)) + if(4+4+(2+2+2))) = 27 \cdot 8 = 216$ cicluri.
Numărul minim de cicluri de tact: $8 + (2+2) = 12$ cicluri.

Problema 10: Doar o procedură

Se consideră procedura RunMe(x, p, a) unde **x** și **p** sunt numere întregi și **a** număr natural.

Cerințe: $x, p \in \mathbb{Z}, a \in \mathbb{N}$

```
procedura RunMe(x, p, a)
    afișează x, " "
    dacă p > 0 atunci
        dacă x > 0 atunci
            RunMe(x − a, p, a)
            afișează x, " "
    altfel
        dacă x < 0 atunci
            RunMe(x + a, p, a)
            afișează x, " "
```

Precizați care dintre următoarele afirmații sunt adevărate:

Ciornă, marcați 1-2 răspunsuri:
(1) (2) (3) (4) (5)

① Pentru apelul RunMe$(9,7,4)$, se va afișa șirul `9␣5␣1␣-3␣1␣5␣9` ✔

② Pentru apelul RunMe$(6,-3,3)$, se va afișa șirul `-6␣-3␣0␣3␣6`

③ Pentru apelul RunMe$(12,3,4)$, se va afișa șirul `12␣8␣4␣0␣-4␣-8␣-12`

④ Pentru apelul RunMe$(-8,-3,3)$, se va afișa șirul `-8␣-5␣-2␣1␣-2␣-5␣-8` ✔

⑤ Pentru apelul RunMe$(6,-3,3)$, se va afișa șirul `6␣3␣0␣-3␣-6`

Explicații

Șirurile de valori afișate trebuie să înceapă cu valoarea inițială a lui x și să fie simetrice în raport cu elementul din centru (prima valoare nulă sau negativă dacă $p > 0$, respectiv prima valoare nulă sau pozitivă dacă $p < 0$) întrucât aceeași valoare se afișează atât la apelul funcției recursive cât și la ieșirea din apel. Deci răspunsurile corecte sunt: pentru $x = 9$, $p = 7$, $a = 4$ se obține șirul `9␣5␣1␣-3␣1␣5␣9`, iar pentru $x = -8$, $p = -3$, $a = 3$ se obține șirul `-8␣-5␣-2␣1␣-2␣-5␣-8`

Problema 11: Rezultat simplu

Se consideră funcția RunMe, de mai jos

Cerințe: $k \in \mathbb{N}$

```
funcția RunMe(k)
    i, rezultat ← 0
    pentru i ← 0 to k execută
        dacă i mod 3 = 1 atunci
            rezultat ← rezultat + i
        altfel
            rezultat ← rezultat + 1
    întoarce rezultat
```

Precizați care dintre următoarele afirmații sunt adevărate:

① Pentru apelul RunMe(8), funcția va returna 18 ✔

② Pentru apelul RunMe(0), funcția va returna 1 ✔

③ Pentru apelul RunMe(5), funcția va returna 5

④ Pentru apelul RunMe(9), funcția va returna 18

⑤ Pentru apelul RunMe(4), funcția va returna 8 ✔

Ciornă, marcați 1-3 răspunsuri:
① ② ③ ④ ⑤

Explicații

Valorile returnate pentru k variind de la 0 la 9 sunt: $1, 2, 3, 4, 8, 9, 10, 17, 18, 19$, deoarece la rezultat se adaugă 1 pentru toate valorile parcurse, cu excepția lui 4 și 7 pentru care se adaugă 4 respectiv 7.
Deci pentru $k = 0$ obținem 1, pentru $k = 8$ obținem 18, iar pentru $k = 4$ obținem 8.

Problema 12: Run it once

Se consideră algoritmul de mai jos

```
funcția RunMe(i, n)
    dacă i = 0 atunci
        întoarce 0
    altfel
```

```
    dacă (n mod 2) = 0 atunci
        întoarce RunMe(i + 1, n) + n
    altfel
        întoarce RunMe(i − 1, n) + n
```

Care sunt valorile de start pentru care execuția algoritmului nu se termină niciodată?

Ciornă, marcați 1-3 răspunsuri: (1) (2) (3) (4) (5)

① $i \leftarrow 2, n \leftarrow 7$
② $i \leftarrow 3, n \leftarrow 4$ ✔
③ $i \leftarrow 5, n \leftarrow 0$ ✔
④ $i \leftarrow 3, n \leftarrow 3$
⑤ $i \leftarrow 2, n \leftarrow 6$ ✔

Explicații

Algoritmul nu se termină niciodată (în pseudocod) atunci când valoarea lui n este pară. Evident răspunsurile corecte sunt: $i \leftarrow 3, n \leftarrow 4$, $i \leftarrow 5, n \leftarrow 0$, $i \leftarrow 2, n \leftarrow 6$.

Problema 13: Run the vector

Se consideră algoritmul RunMe(a, n) unde **n** este număr natural ($2 \leq n \leq 1\,000$) și **a** un vector de **n** numere întregi ($a[1], a[2], ..., a[n]$).

Cerințe: $n \in \mathbb{N}, 2 \leq n \leq 1000$

```
funcția RunMe(a, n)
    b, i ← 1, j ← n
    cât timp j > i execută
        cât timp a[i] mod 2 ≠ 0 ȘI i ≤ n execută
            i ← i + 1
        cât timp a[j] mod 2 = 0 ȘI j ≥ 1 execută
            j ← j − 1
        dacă i < j atunci
            a[i] ← a[i] + a[j]
            a[j] ← a[i] − a[j]
            a[i] ← a[i] − a[j]
```

Ce efect are algoritmul asupra vectorului **a**:

Ciornă, marcați 1-3 răspunsuri: (1) (2) (3) (4) (5)

① plasează elementele impare în prima parte a vectorului ✔
② ordonează crescător vectorul
③ dacă toate elementele sunt pozitive, nu are niciun efect
④ plasează elementele pozitive în a doua parte a vectorului
⑤ dacă toate elementele sunt pare, nu are niciun efect ✔

Explicații

În cazul algoritmului de mai sus, i este incrementat cu valoarea 1 până se găsește un element par sau i devine mai mare decât n, iar j este decrementat cu valoarea 1 până se găsește un element impar sau j devine mai mic decât 1. În cazul în care i este încă mai mic decât j se vor interschimba elementele de pe pozițiile i și j, un element impar ajunge în fața unui element par. La finalul algoritmului toate elementele impare vor fi în prima parte a vectorului, iar dacă toate elementele sunt pare, j este drecrementat până la 0 și algoritmul nu are niciun efect.

Problema 14: Odd enough

Se consideră algoritmul RunMe(n) de mai jos, unde **n** este număr întreg.

Cerințe: $n \in \mathbb{Z}$

```
funcția RunMe(n)
    dacă n ≤ 0 atunci
        întoarce 2
    întoarce RunMe(n − 2) + 5
```

Precizați care dintre următoarele afirmații sunt adevărate:

① Pentru apelul RunMe(6), algoritmul returnează 17 ✔
② Pentru apelul RunMe(5), algoritmul returnează 17 ✔
③ Pentru apelul RunMe(10), algoritmul returnează 15
④ Pentru apelul RunMe(−1), algoritmul returnează 2 ✔
⑤ Pentru apelul RunMe(2), algoritmul returnează 15

Ciornă, marcați 1-3 răspunsuri: ① ② ③ ④ ⑤

Explicații

În cazul algoritmului de mai sus, funcția este apelată recursiv pentru $(n - 2)$ care ne oferă numărul de pași până când $n = 0$ sau $n < 0$. Când $n = 0$ sau $n < 0$ se obține valoarea 2 la care se adaugă valoarea 5 pentru fiecare pas. Astfel răspunsurile corecte sunt: pentru $n = 6$, se obține 17, pentru $n = 5$, se obține $17 (2 + 5 + 5 + 5)$, iar pentru $n = -1$, se obține 2.

Problema 15: Un vector

Se consideră un vector **a** cu **n** numere întregi (**a**[1], **a**[2], ..., **a**[n]), numărul natural **n** ($1 \leq n \leq 1\,000$) și numărul întreg **x**. Care din următoarele secvențe de cod returnează poziția cu indicele maxim la care se găsește valoarea **x** în vectorul **a**, sau returnează -1 dacă **x** nu se găsește în **a**?

①
```
i ← n
cât timp i ≥ 1 execută
    i ← i − 1
    dacă a[i] = x atunci
        întoarce i
întoarce −1
```

②
```
i ← n, b ← −1
cât timp i ≥ 1 execută
    dacă a[i] = x atunci
        b ← i
    i ← i − 1
întoarce b
```

③ ✔
```
i ← 1, b ← −1
cât timp i ≤ n execută
    dacă a[i] = x atunci
        b ← i
    i ← i + 1
întoarce b
```

④ ✔
```
i ← n
cât timp i ≥ 1 execută
    dacă a[i] = x atunci
        întoarce i
    i ← i − 1
întoarce −1
```

⑤ Niciuna dintre secvențele de mai sus

Ciornă, marcați 1-3 răspunsuri: ① ② ③ ④ ⑤

Explicații

În cazul enunțului de mai sus răspunsurile corecte sunt (3) și (4). În cazul (3) vectorul este parcurs de la început și de fiecare dată când un element este egal cu x, actualizează poziția elementului în variabila b care este returnată la sfârșit, deci este returnată ultima poziție pe care se află x. În cazul (4) vectorul este parcurs de la sfârșit iar când este găsit x este returnată poziția lui.
Varianta (1) este incorectă pentru că nu analizează pe $a[n]$, iar varianta (2) este incorectă pentru că modifică pe b la fiecare intâlnire a lui x, deci va întoarce prima poziție pe care se află x.

Problema 16: ∞

Se consideră algoritmul:

Cerințe: $n \in \mathbb{N}$

```
funcția RunMe(n)
    dacă n = 0 atunci
        întoarce 0
    altfel
        dacă (n mod 2) ≠ 0 atunci
            întoarce RunMe(n − 1)
    întoarce RunMe(n − 1) + n
```

Precizați care dintre afirmațiile următoare sunt corecte:

1. RunMe(111) = 3 078
2. RunMe(777) = 150 932 ✔
3. Funcția RunMe() poate returna numere impare
4. Funcția RunMe() poate returna numere pare ✔
5. Funcția RunMe() poate intra uneori în ciclu infinit

Ciornă, marcați 1-3 răspunsuri:
(1) (2) (3) (4) (5)

Explicații

Functia calculează suma numerelor pare mai mici decat n (adică $2\sum_{i=1}^{\lfloor n/2 \rfloor} i$)
Pentru `RunMe(777)` obținem suma numerelor pare de la 0 la 776, deci avem $\lfloor n/2 \rfloor = 388$, în concluzie `RunMe(777)` $= 388 * 389 = 150,932$
Pentru `RunMe(111)` obținem suma numerelor pare de la 0 la 111, deci avem $\lfloor n/2 \rfloor = 55$, în concluzie `RunMe(111)` $= 55 * 56 = 3,080$

Problema 17: *Niște șiruri*

Fiind dat algoritmul de mai jos, unde argumentele $s1, s2$ transmise funcției sunt șiruri de caractere (care sunt indexate începând cu poziția 1), iar n, m sunt numere naturale:

Cerințe: $n, m \in \mathbb{N}$

```
funcția RunThat(s1, s2, n, m)
    dacă n > 0 ȘI m > 0 atunci
        dacă s1[n] ≠ s2[m] atunci
            întoarce n + m
        altfel
            întoarce RunThat(s1, s2, n − 1, m − 1)
```

```
    întoarce 0
```

Precizați care dintre afirmațiile următoare sunt corecte:

Ciornă, marcați 1-3 răspunsuri: (1) (2) (3) (4) (5)

(1) Funcția poate fi folosită pentru compararea a două șiruri
(2) RunThat("MERGE!", "SUPER BETON!", 6, 12) = 16 ✔
(3) Funcția RunThat() returnează uneori și valoarea 0 ✔
(4) Funcția RunThat() poate intra uneori în ciclu infinit
(5) Funcția RunThat() poate avea comportament nedefinit pentru anumite valori ale parametrilor ✔

Explicații

Dacă șirurile sunt de lungimi egale funcția va returna 0 (varianta 3).
RunThat("MERGE!", "SUPER BETON!", 6, 12) = 5 + 11 = 16 (varianta 2) unde 5 și 11 reprezintă pozițiile literelor care nu corespund pornind de la sfârșitul șirurilor.
Dacă n sau m este mai mare decât lungimea șirului corespunzător, atunci funcția RunThat() poate avea comportament nedefinit (varianta 5).

Problema 18: This and that

Se consideră algoritmul RunThat(n), unde $1 \leq n \leq 1\,000$.

Cerințe: $n \in \mathbb{N}, 1 \leq n \leq 1000$

```
funcția RunThat(n)
    dacă n < 2 atunci
        întoarce n
    a ← 0; b ← 1; c ← 0
    pentru i ←2 to n execută
        c ← a + b; a ← b; b ← c
    întoarce c
```

Precizați care dintre următoarele afirmații sunt adevărate:

Ciornă, marcați 1-3 răspunsuri: (1) (2) (3) (4) (5)

(1) Pentru apelul RunThat(5), algoritmul returnează valoarea 5 ✔
(2) Pentru apelul RunThat(20), algoritmul returnează valoarea 2765
(3) Pentru apelul RunThat(15), algoritmul returnează valoarea 600
(4) Pentru apelul RunThat(10), algoritmul returnează valoarea 55 ✔
(5) Pentru apelul RunThat(8), algoritmul returnează valoarea 21 ✔

Explicații

Valorile calculate de algoritm corespund șirului definit de următoare relație de recurență: $s_n = s_{n-1} + s_{n-2}, s_1 = s_2 = 1$

Problema 19: Eficiență maximă

Algoritmul de mai jos își propune să însumeze într-un mod **eficient** divizorii numărului n, însă unele instrucțiuni sunt incomplete:

Cerințe: $n \in \mathbb{N}, n > 0$

```
funcția RunMe(n)
    sum ← 0
```

```
pentru ⟨1⟩ execută
    dacă ⟨2⟩ atunci
        sum ← sum + d
        dacă ⟨3⟩ atunci
            sum ← sum + n/d
întoarce sum
```

Completați instrucțiunile care lipsesc astfel:

Ciornă, marcați 1-3 răspunsuri: (1) (2) (3) (4) (5)

(1) ⟨1⟩ cu $d \leftarrow 1$ to $sqrt(n)$; ⟨2⟩ cu $n \mod d = 0$; ⟨3⟩ cu $d \neq n/d$ ✔

(2) ⟨1⟩ cu $d \leftarrow 1$ to n; ⟨2⟩ cu $d \mod n = 0$; ⟨3⟩ cu $d \neq 0$

(3) ⟨1⟩ cu $d \leftarrow 1$ to $sqrt(n)$; ⟨2⟩ cu $n \mod d \neq 0$; ⟨3⟩ cu $n \mod n/d = 0$

(4) ⟨1⟩ cu $d \leftarrow 1$ to n; ⟨2⟩ cu $n \mod d = 0$; ⟨3⟩ cu $d \neq n/d$

(5) ⟨1⟩ cu $d \leftarrow 1$ to $sqrt(n)$; ⟨2⟩ cu $d \mod n = 0$; ⟨3⟩ cu $n \mod n/d = 0$

Explicații

Răspunsul corect este: ⟨1⟩ cu $d \leftarrow 1$ to $sqrt(n)$; ⟨2⟩ cu $n \mod d = 0$; ⟨3⟩ cu $d \neq n/d$.

Problema 20: Este produs?

Algoritmul de mai jos își propune să înmulțească două numere naturale a și b, cu $a \geq 0$ și $b \geq 0$.

Cerințe: $a, b \in \mathbb{N}, a \geq 0, b \geq 0$

```
1: funcția RunMe(a, b)
2:     dacă b = 0 atunci
3:         întoarce 0
4:     dacă a mod 2 = 0 atunci
5:         întoarce RunMe(2 * a, b/2)
6:     altfel
7:         întoarce RunMe(2 * a, b/2 + a)
```

S-au strecurat însă două greșeli. Găsește-le și corectează-le!

Ciornă, marcați 1-3 răspunsuri: (1) (2) (3) (4) (5)

(1) trebuie modificat testul din linia 4 în $a \mod 2 \neq 0$

(2) trebuie modificat testul din linia 4 în $b \mod 2 = 0$ ✔

(3) trebuie modificat apelul din linia 5 în $\text{RunMe}(2 * a, b/2) + a$

(4) trebuie modificat apelul din linia 7 în $\text{RunMe}(2 * a, b/2) + a$ ✔

(5) trebuie modificat apelul din linia 7 în $\text{RunMe}(2 * a, b/2)$

Explicații

Răspunsurile corecte sunt: trebuie modificat testul din linia 8 în $\text{RunMe}(2 * a, b/2) + a$ și trebuie modificat testul din linia 5 în $b \mod 2 = 0$

.