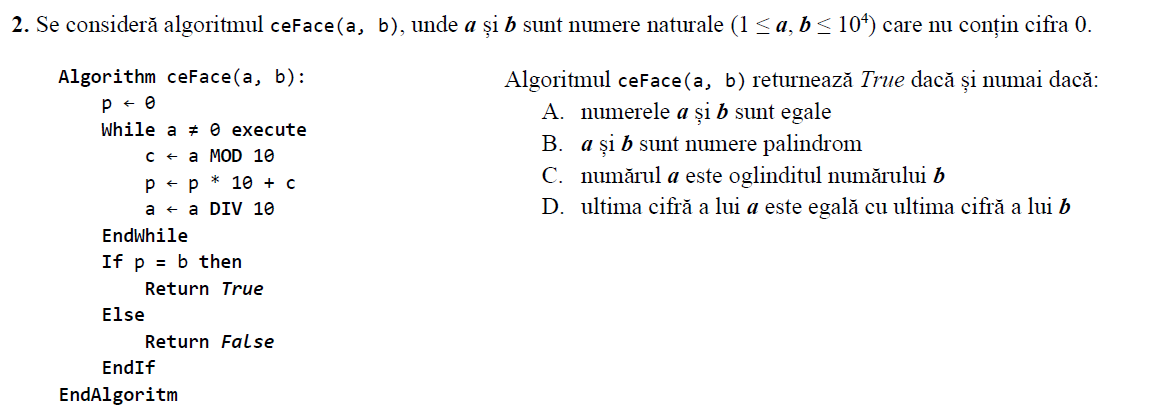


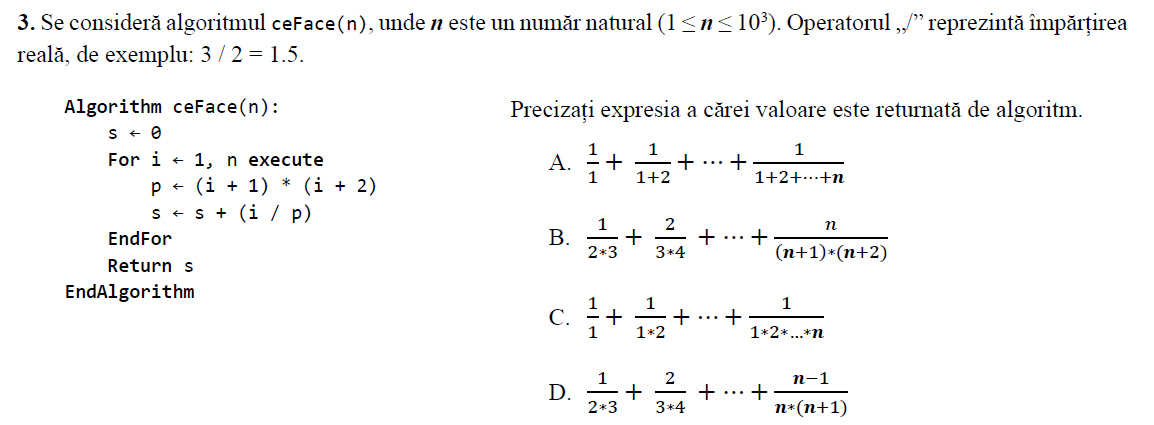
Răspuns: A, B

Numără câte dintre prefixele numerelor sunt divizibile cu 3.

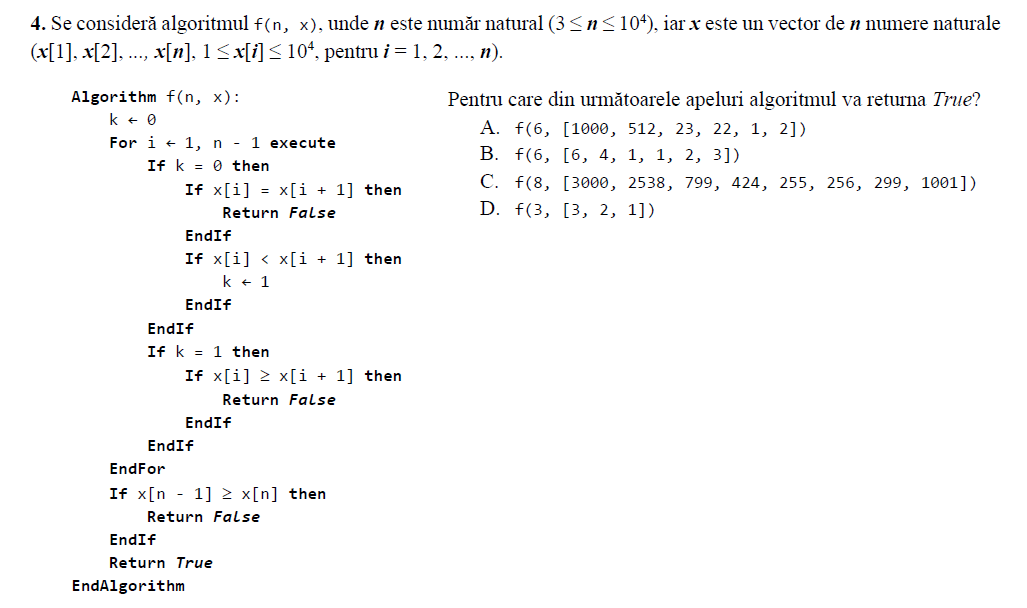


Răspuns: C

Se vede că funcția formează în p oglinditul l ui a și verifică dacă este egal cu b.

Atenție la nuanța lui "dacă și numai dacă" (fără de care ar fi fost corect și b-ul)

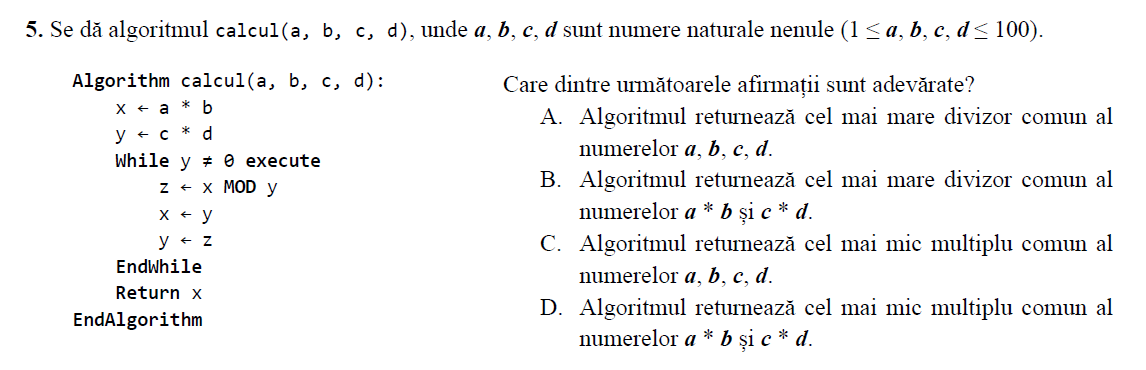
Răspuns: B (urmează exact forma dată de algoritm)



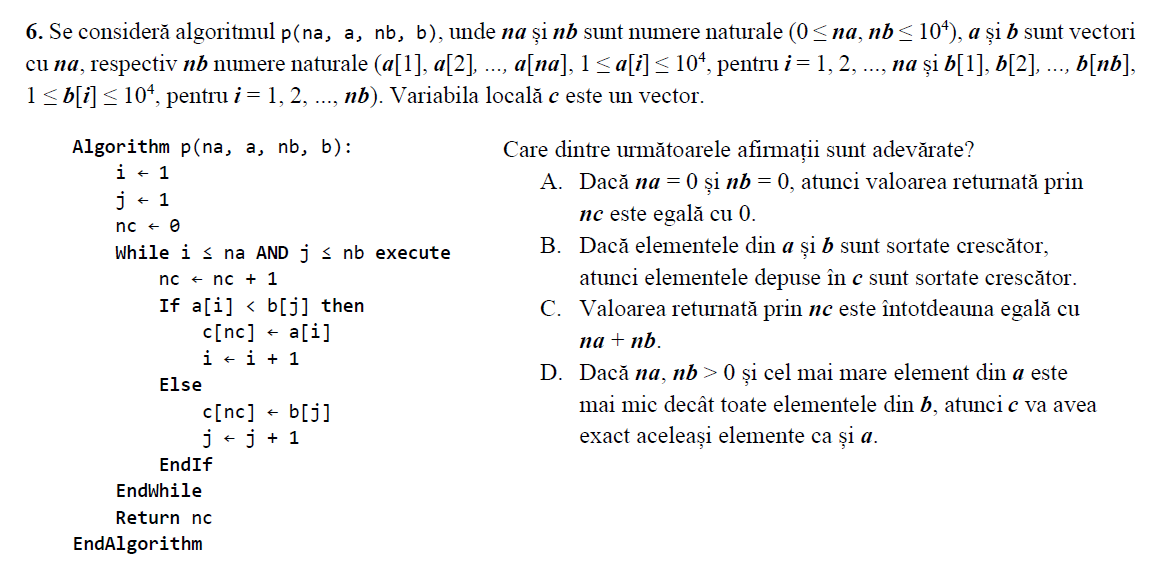
Răspuns: A, C

Subprogramul verifică dacă șirul are aspect de vale - adică are un minim unic, care nu se află nici pe prima nici pe ultima poziție, până la minim este strict descrescător și apoi strict crescător.

Excepție: Mai dă TRUE și dacă este strict crescător



Răspuns: B (mergem direct pe definiție)



Răspuns: A, B, D

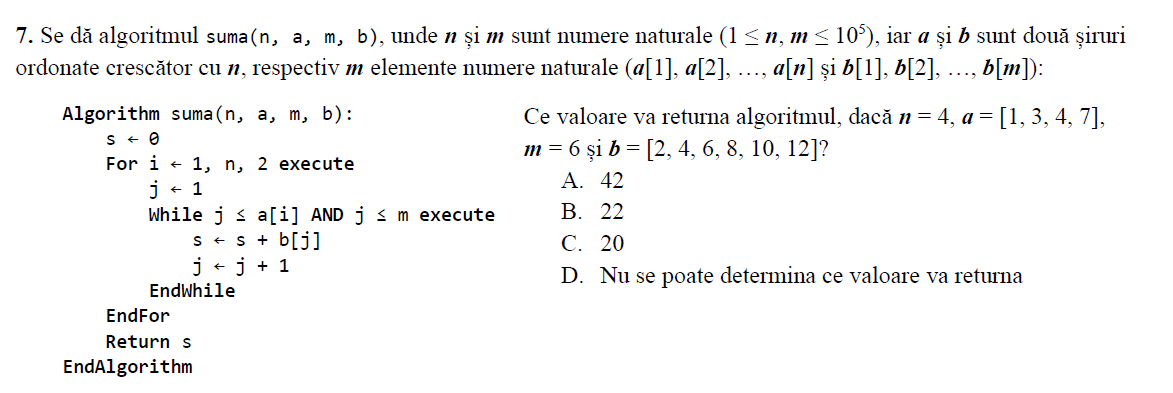
Algoritmul face prima parte de la interclasare (până se termină exact unul dintre șiruri).

A - corect pt. că NU intră pe while

B - corect - bazat pe faptul că alg. este derviat de la interclasare

C - FALS - pt. că while-ul se termină în momentul în care UNUL dintre șiruri este parcurs (nu și celălalt)

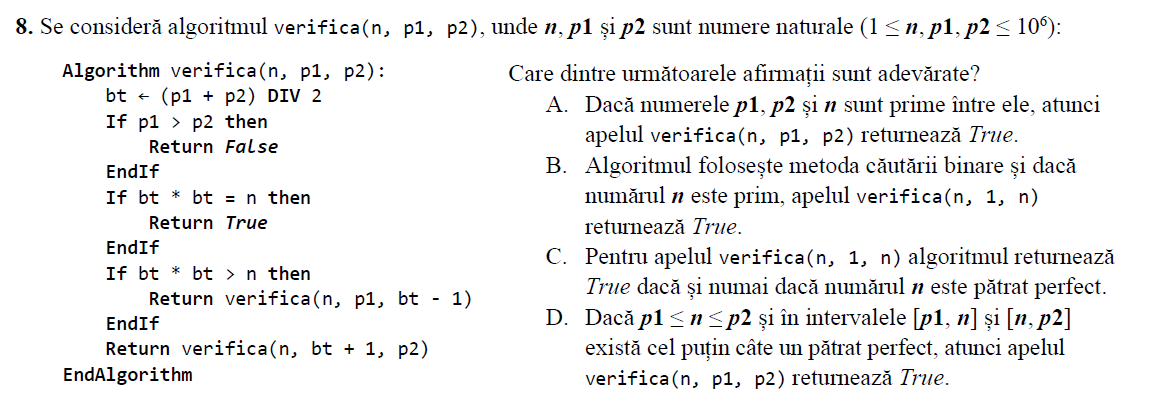
D - ADEVĂRAT - dacă maximul din a este mai mic ca minimul din b, înseamnă de fapt că oricare din a este mai mic ca oricare din b, deci if-ul de pe repetitivă ne ține pe ramura then până se termină tot șirul a, după care iese.



Răspuns: B - trebuie urmărit cu atenție. Practic există doar pașii i=1 și i=3.

Pt. i=1 j face doar pasul j=1

i=2 j face pașii 1,2,3,4

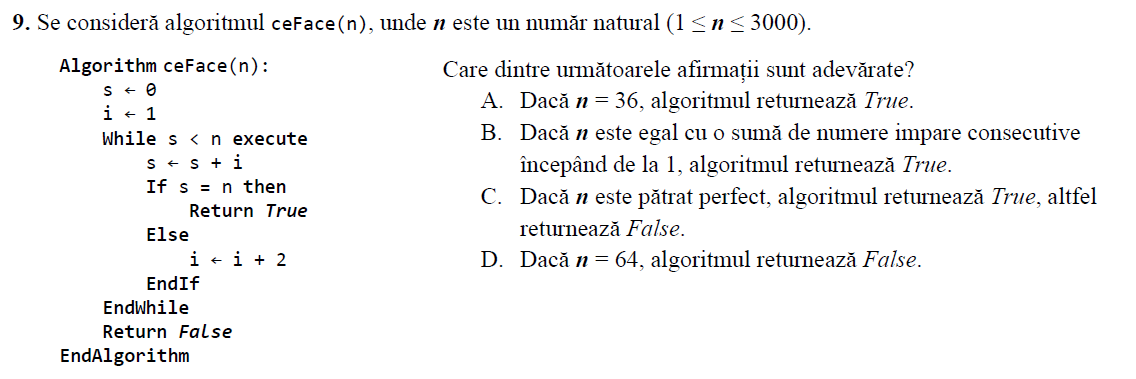


Răspuns: C

Algoritmul este de tip căutare binară. Mai precis întoarce TRUE doar dacă n este pătratul unui număr din intervalul [p1,p2]. Altfel, întoarce FALSE

De exemplu verifica(25,4,6) sau verifica(25,1,10) sau verifica(25,1,25) este TRUE

Pe când verifica(25,6,20) sau verifica(25,15,40) este FALSE



Răspuns: A,B,C

Dacă ne-apucăm să verificăm A-ul, calculăm suma:

1+3+5+7... cât timp aceasta este < 36.

Pe parcursul calculelor ne prindem că:

1+3=4

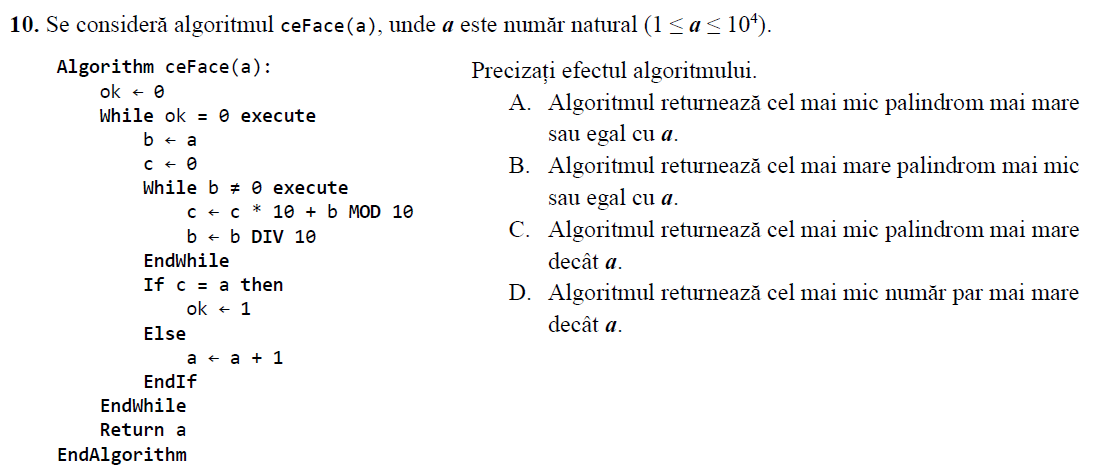
4+5=9

9+7=16

16+8=25

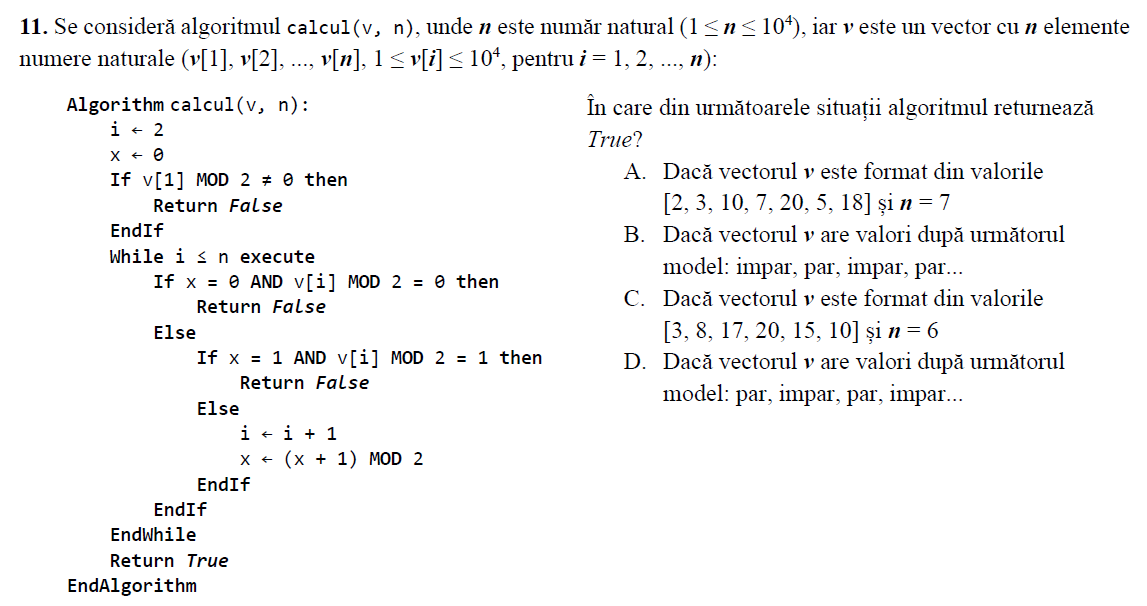
25+9=36 De fapt TOATE rezultatele parțiale sunt pătrate perfecte.

Deci funcția se bazează pe proprietatea că orice sumă de numere impare consecutive care începe de la 1 este un pătrat perfect



Răspuns: A

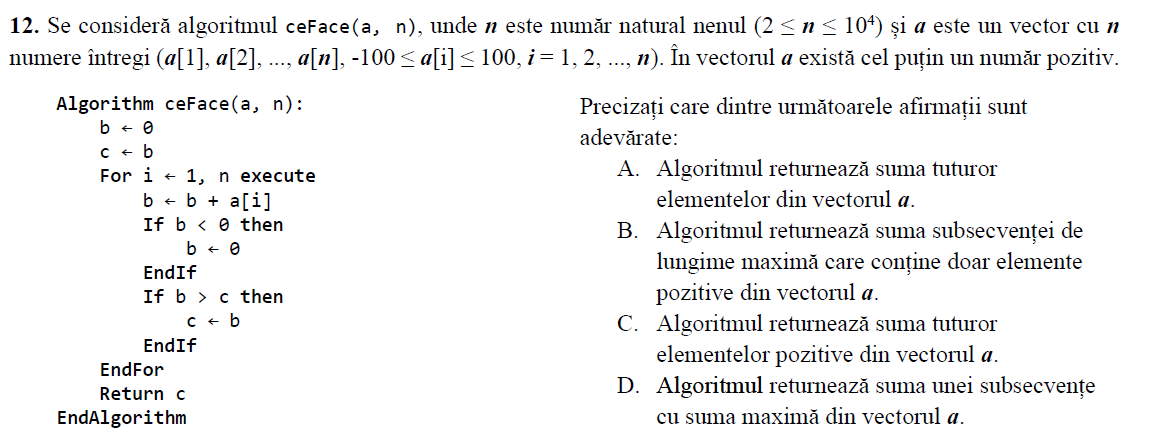
Observăm că, față de a-ul dat, pleacă la plimbare crescător. Deci numărul determinat nu va mi mic decât a. Se vede și că în c se formează oglinditul lui a, deci condiția de oprire este să fie palindrom. Ca să ne dăm seama corect care dintre A și C este răspunsul bun, luăm un număr care este deja palindrom. Dacă funcția ni-l dă chiar pe el (și de fapt așa este) răspunsul este A.



Răspuns: A, D

Obs: x de fapt alternează la fiecare pas: 0, 1, 0, 1, ...

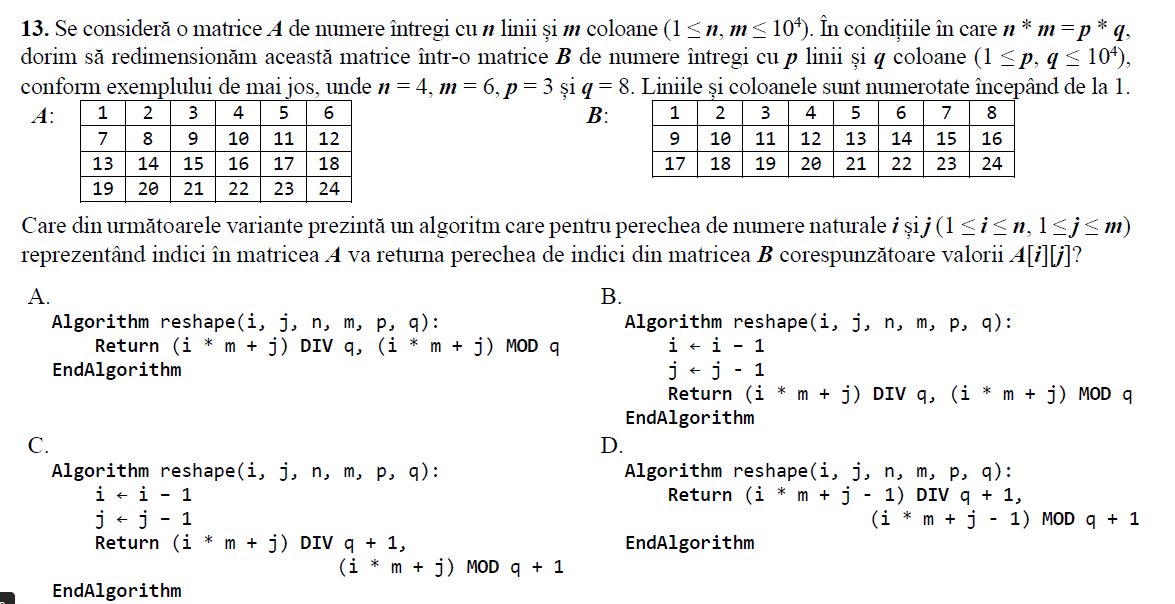
Și funcția, în timp ce crește x-ul crește și i-ul și verifică dacă pe poz. impare avem numere pare și invers.



Răspuns: D

Algoritmul face sume parțiale (deci de secvențe), resetând sumele când devin negative, lucru de înțeles, pt. că dacă după o sumă negativă dăm de un număr pozitiv, e mai convenabil să-l alegem pe acesta singur decât să-l adunăm la ceva negativ.

Problema este trivială (devine banală) dacă vectorul are DOAR numere pozitive.



Răspuns: C

putem lua un element din matricea A și calcula indicii - trebuie să obținem indicii din B pt. același număr.

De exemplu: dacă luăm 20-ul din matricea A:

i=4 j=2 știm că n=4 m=6 p=3 q=8

să verificăm:

A) )new\_i=(4\*6+2)div 8 = 26 div 8 =3 asta ar fi bună

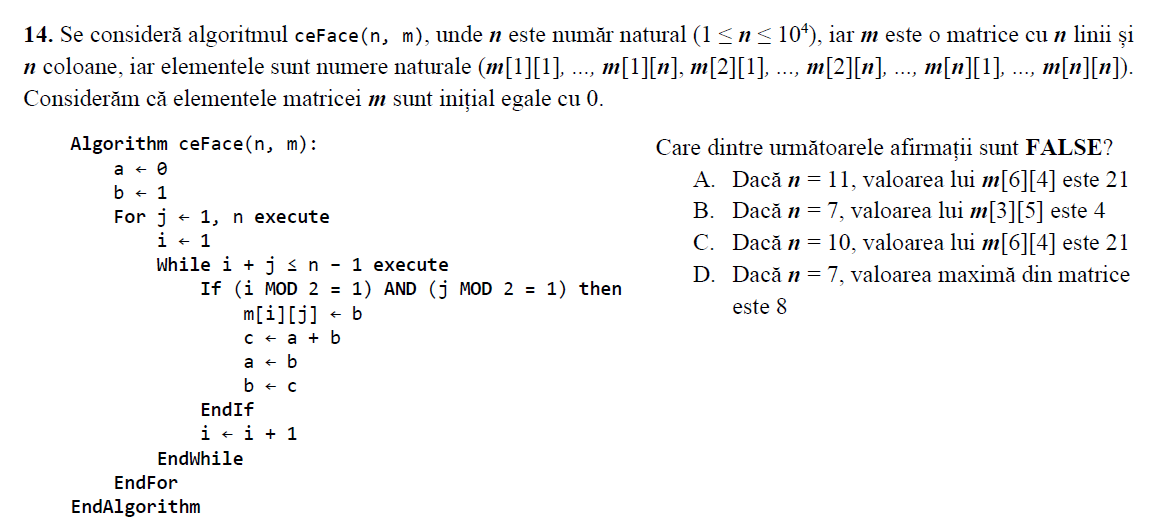
new j = (4\*6+2)mod 8 = 2 NU e bună, tre' să dea 4

B) i=3;j=1 -> new\_i=(3\*6+1)div 8 = 19 div 8 = 2 deci NU

C) i=3;j=1 -> new\_i=(3\*6+1)div 8+1 = 19 div 8+1 = 3

new\_j=(3\*6+1)mod 8+1 = 19 mod 8 + 1 = 3+4 = 4 deci DA

dacă verificăm și D-ul vedem că nu e bun



Răspuns:

Recunoaștem

c←a+b; a←b; b←c - generarea de termeni din Fibonacci (mai ales că și inițial a=0,b=1)

for j=1,n - parcurgem o matrice pe coloane

i=1 combinat cu while (i+1<=n-1) iar pe while avem i=i+1

înseamnă că fiecare coloană este parcursă în jos până cu diagonale deasupra celei secundare.

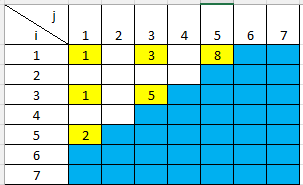
În fine, faptul i mod 2=1 și jmod 2=1 înseamnă că doar pe linii imparae și coloane impare dintre acestea ni se pun termeni din Fibonacci.

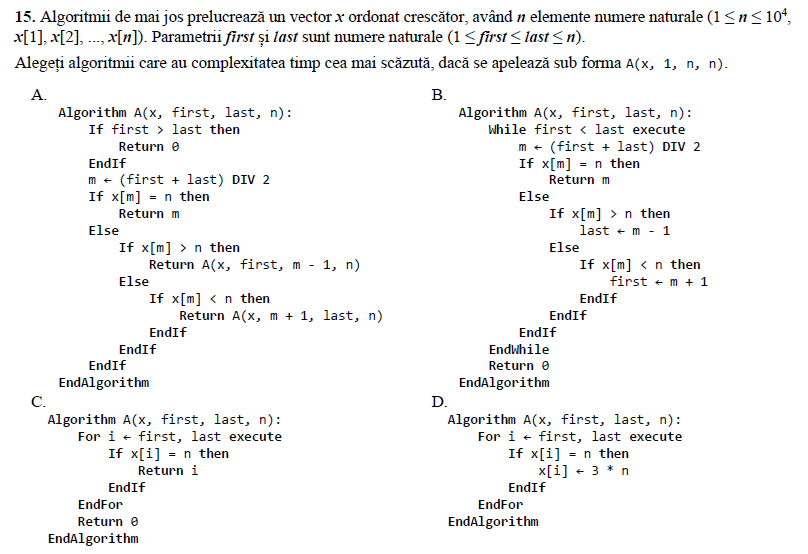
Datorită DOAR acestei ultime restricții => pe indici pari matricea rămâne 0.

Pică imediat variantele A și C.

Din faptul că TOȚI termenii care se pun în matrice sunt din Fibo, ne pică imediat și B-ul (4 NU e Fibo)

Ca să facem o verificare ultra riguroasă la D) cam așa arată matricea:

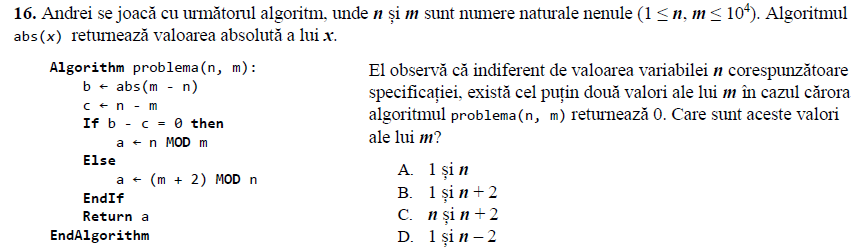




Răspuns: A,B

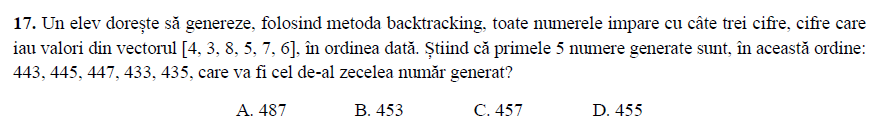
Cele două scrieri reprezintă varianta recursivă respectiv nerecursivă a aceluiași algoritm: căutarea binară. Știm că acest algoritm are complexitate logaritmică.

Pe când C și D conțin câte un for, deci sunt liniari, deci mai mult.



Răspuns: A

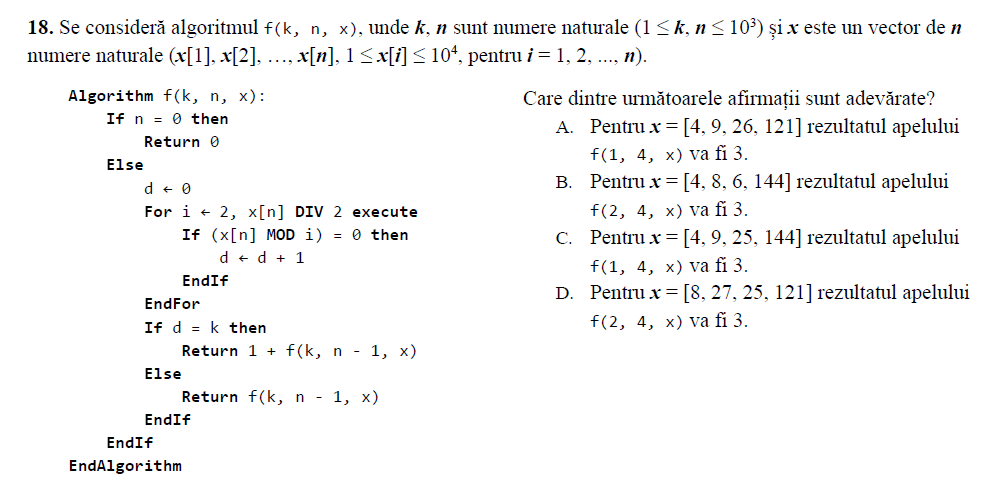
(verificăm rând pe râdn pt. m=1, m=n, m=n+2, m=n-2 și vedem că doar primele 2-s bune)



Răspuns: B

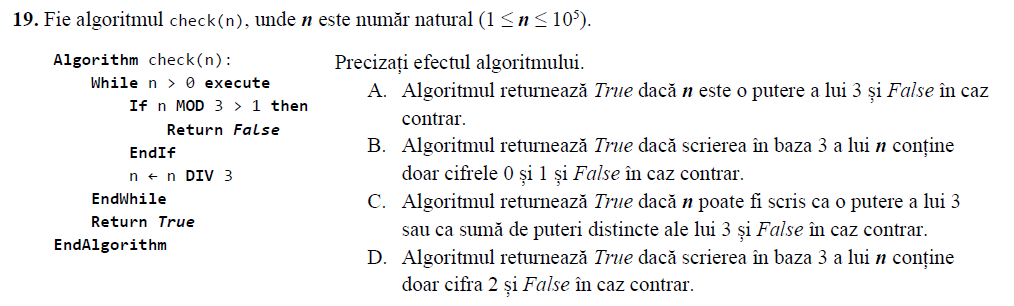
(se generează produse carteziene, iar numerele le luăm în ordinea dată în vector,

numerele generate de la ultimul dat până la soluție fiind: 437, 483, 485, 487, **453**)



Răspuns: A, C

Află numărul de elemente din vector care au exact k divizori proprii (fără 1 și el însuși)



Răspuns: B,C

Algoritmul reprezintă de fapt transformarea lui n în baza 3.

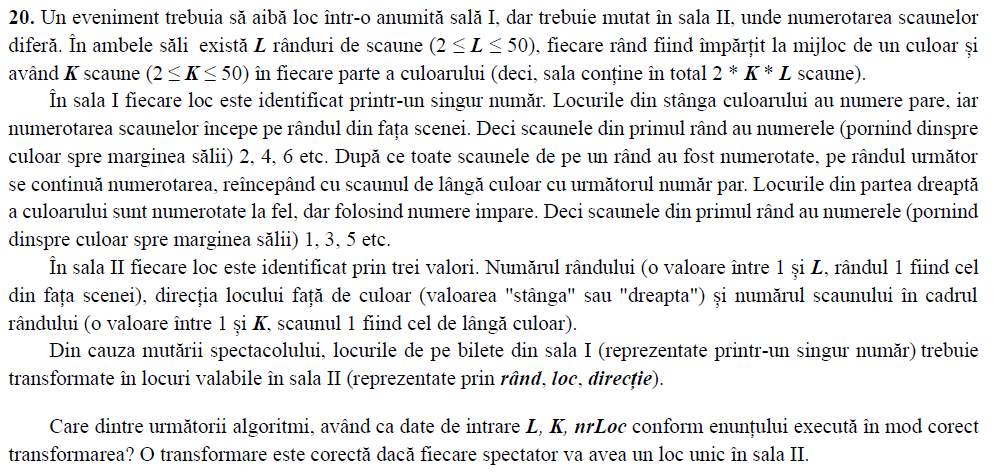
Mai precis, n MOD 3 reprezintă tot câte o cifră a lui n când e scris în baza 3.

Din acest motiv, B este un răspuns care ne dă imediat că e adevărat.

Apoi, dacă ne raportăm la scrierea unui număr în baza 3, aceasta este:

abcd(3) = a⋅33+ b⋅32+c⋅31+d⋅30, adică o sumă de puteri ale lui înmulțite cu ceva.

Dat fiind că ceva-urile ălea sunt doar 0 sau 1 ⇒ și C este adevărat



Răspuns: A

Cam cum arată sălile:

Să le figurăm pe ambele pt. L=3, K=4

I:

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

II:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| r=1,dir=st  loc=4 | r=1,dir=st  loc=3 | r=1,dir=st  loc=2 | r=1,dir=st  loc=1 | Le couloir | r=1,dir=dr  loc=1 | r=1,dir=dr  loc=2 | r=1,dir=dr  loc=3 | r=1,dir=dr  loc=4 |
| r=2,dir=st  loc=4 | r=2,dir=st  loc=3 | r=2,dir=st  loc=2 | r=2,dir=st  loc=1 | r=2,dir=dr  loc=1 | r=2,dir=dr  loc=2 | r=2,dir=dr  loc=3 | r=2,dir=dr  loc=4 |
| r=3,dir=st  loc=4 | r=3,dir=st  loc=3 | r=3,dir=st  loc=2 | r=3,dir=st  loc=1 | r=3,dir=dr  loc=1 | r=3,dir=dr  loc=2 | r=3,dir=dr  loc=3 | r=3,dir=dr  loc=4 |

În coduri constatăm o diferențiere la faptul că nrloc se divide cu 8.

Ceea ce are sens la locurile 8, 16, 24. La C nu apare -> C este fals

Pentru celelalte, dacă luați loc=1 în sala inițială și loc=2 putem vedea că doar A-ul este corect