REZOLVARI - CONSULTATII

3. Numarul total de valori dintr-un interval [a, b], atunci cand pasul este k, se calculeaza folosind formula ((b−a)/ k)+1. In algoritmul dat, daca a > b, se face o interschimbare pe biti folosind operatorul XOR, iar pasul nu este k, ci 2k, deoarece, in fiecare iteratie, a se incrementeaza cu k, iar b se decrementeaza cu k. Prin urmare, numarul total de valori va fi (b-a)/(2K) + 1. La punctul B, se afiseaza 12 valori, aplicand formula. La punctul C, se afiseaza corect valorile. La punctul D, ultima valoare care se afiseaza este mai mare decat (a+ b) / 2. Asadar, afirmatiile care nu sunt adevarate sunt B si D.
Rapuns corect: B, D

11. Pentru (0+8) / (4-2) + 17 mod 3 = 8/2 + 2 = 4+2 = 6, unde am calculat mai intai 16 mod 4 = 0, $4 * 2 = 8$, $16 \div 4 = 4$, iar apoi 17 mod 3 = 2.
Raspuns corect: C

45. Expresiile corecte sunt cele care verifica simultan ca x sa fie multiplu de 5 si sa apartina intervalului (a, b]. Expresia A realizeaza aceasta verificare utilizand negatia si operatorii logici pentru a exclude valorile din afara intervalului.
Raspuns corect: A

20. A. Expresia are valoarea False. B. Expresia are valoarea True. C. Expresia are valoarea False. Expresia are valoarea True.
Raspuns corect: A C

69. Se transforma termenii expresiei E in baza 10, se calculeaza rezultatul expresiei, apoi se fac conversiile in bazele indicate in variantele de raspuns.
Raspuns corect: B C D

74. Algoritmul calculeaza numarul de cifre 0 din reprezentarea in baza 6 a numarului n.
Raspuns corect: A D

80. Algoritmul afiseaza resturile succesive ale impartirii lui n la 2 in ordinea lor, dar pentru a obtine reprezentarea binara corecta, aceste resturi trebuie citite in ordine inversa. Astfel, afirmatiile A si B sunt corecte. D este corect deoarece complexitatea algoritmului este O(log n).
Raspuns corect: A B D

87. Algoritmul returneaza cifra cu cel mai mare rest la impartirea la 4 din n. Daca exista mai multe cifre cu acelasi rest, este selectata cifra cea mai semnificativa. A: Corect, pentru n = 12569, algoritmul returneaza 2. B: Fals, pentru n = 783031, algoritmul returneaza 7. C: Corect, pentru n = 2024 si n = 2025, algoritmul returneaza 2. D: Fals, algoritmul nu returneaza prima cifra divizibila cu 4.
Raspuns corect: A C

82. Algoritmul determina si afiseaza cel mai mic numar din intervalul [a, b] cu numar maxim de divizori, numarul de divizori al acestuia si numarul de numere cu aceasta proprietate. A: Se afiseaza 200 12 1. B: Numarul maxim de divizori este 4 pentru numerele {6, 8, 10}.
Raspuns corect: B, C

89. Algoritmul switch(a, b) returneaza CMMDC dintre a si b, iar case(a, b) returneaza CMMMC dintre a si b. Doua numere prime intre ele au CMMDC egal cu 1. Astfel, variantele corecte sunt A, B, D.
Raspuns corect: A B D

98. Algoritmul algo calculeaza un numar format printr-o secventa de operatii asupra cifrelor din n, tinand cont de conditia specificata si decrementand k cand conditia nu este indeplinita.
Raspuns corect: A C

103. A. Adevarat. B. Fals, instructiunea ”k ← k∗b” trebuie executata dupa instructiunea ”rez ← rez + (nr mod 10) ∗ k”. C. Fals, apelul recursiv trebuie inmultit cu b nu parametrul b din apel. D. Adevarat.
Raspuns corect: A D

6 Decembrie 2025; orele 12; Asist. Univ. Drd. Coste Claudia Ioana

117. Functia rearanjeaza cifrele numarului n astfel: cifrele impare apar in ordine directa, iar cifrele pare in ordine inversa, pastrandu-si pozitiile relative.
Raspuns corect: B C

123. Algoritmul verifica daca n este patrat perfect folosind metoda diferentelor consecutive. Un numar n este patrat perfect daca diferenta dintre orice doua patrate perfecte consecutive este constanta si egala cu 2d + 1, unde d reprezinta numarul patrat. De exemplu: 1 = 1, 4 = 1 + 3, 9 = 1 + 3 + 5, 16 = 1 + 3 + 5 + 7, 25 = 1 + 3 + 5 + 7 + 9, etc. A. Adevarat. B. Adevarat. C. Adevarat, cel mai mic numar patrat perfect din intervalul [71, 734] este 9^2 , iar cel mai mare este 27^2, deci există 27 − 9 + 1 = 19 valori pentru care se returneaza True. D. Fals, in multimea {73, 9, 442, 841, 576, 962}, numerele patrate perfecte sunt 9 = 3^2 , 841 = 29^2 , 576 = 24^2 , deci se returneaza True pentru 3 valori si False pentru 3 valori.
Raspuns corect: A B C

187. Algoritmul este o functie recursiva care afiseaza pentru fiecare apel valoarea dublului parametrului de intrare urmat de un simbol ”!”, apoi apeleaza recursiv functia pentru fiecare numar de la 1 la n − 1, adaugand simbolul ”*” dupa fiecare apel. Structura rezultata reflecta procesul de recursivitate si ordinea apelurilor.
Raspuns corect: D

186. Algoritmul spirala combina prima si ultima cifra ale unui numar in functie de directia specificata de parametrul sens si aplica recursiv aceeasi operatie pe restul cifrelor.
Raspuns corect: C

Subiectele probelor scrise de informatică, admitere nivel licenţă, Iulie 2016, Subiectul I
Problema TURNURI

```
#include <iostream>
using namespace std;

struct turnuri {
   int nr_max;
   int nr_monede;
};

int nrTurnuri(int n) {
   if (n == 1) {
      return 1;
   } else {
      return 1 + 2 * nrTurnuri(n-1);
   }
}

int nrMonede(int n) {
   if (n == 1) {
      return 1;
   } else {
      return n + 2 * nrMonede(n-1);
   }
}

turnuri calculeaza_turnuri(int inaltime_max) {
   turnuri rez;
   rez.nr_max = nrTurnuri(inaltime_max);
   rez.nr_monede = nrMonede(inaltime_max);
   return rez;
}

int main() {
   turnuri rez = calculeaza_turnuri(3);
   cout << rez.nr_max << ' ' << rez.nr_monede << ' ' << endl;
   rez = calculeaza_turnuri(4);
   cout << rez.nr_max << ' ' << rez.nr_monede << ' ' << endl;
   rez = calculeaza_turnuri(5);
```

```
   cout << rez.nr_max << ' ' << rez.nr_monede << ' ' << endl;
   rez = calculeaza_turnuri(6);
   cout << rez.nr_max << ' ' << rez.nr_monede << ' ' << endl;
   rez = calculeaza_turnuri(7);
   cout << rez.nr_max << ' ' << rez.nr_monede << ' ' << endl;
   rez = calculeaza_turnuri(8);
   cout << rez.nr_max << ' ' << rez.nr_monede << ' ' << endl;

   return 0;
}
```

```
// V1,Subiectul I, problema 1: Turnuri

#include <iostream>
using namespace std;

/*
Explicatie

Turnul de inaltime n este obligatoriu singur (altfel, intre doua turnuri de
inaltime n trebuie sa existe un turn de inaltime n+1, care e prea inalt).

Excluzand turnul de inaltime n, putem avea cel mult 2 turnuri de inaltime n-1
(trei sau mai multe turnuri de inaltime n-1 duc la doua sau mai multe
turnuri de inaltime n)
de o parte si de alta a turnului de inaltime n
(intre ele trebuie sa existe cel putin un turn de inaltime mai mare).

La stanga turnului de inaltime n
se vor construi maximul de turnuri (conform cerintelor problemei)
cu cel mai inalt turn avand inaltimea n-1.
(Similar pentru constructia de la dreapta turnului de inaltime n.)

Obtinem deci urmatoarea constructie recursiva: sirul de turnuri este format
dintr-un turn de inaltime n aflat la mijloc, iar de fiecare parte avem
un sir de turnuri care este identic cu constructia ce conduce la cel mai
inalt turn la inaltime n-1.

Ca urmare, avem recurentele:
  nrTurnuri(n) = 2*nrTurnuri(n-1) + 1
  nrMonede(n) = 2*nrMonede(n-1) + n

Terminarea recurentelor se face la n=1:
  nrTurnuri(1) = 1
  nrMonede(1) = 1

Implementarea se poate face recursiv;
recursia poate fi desfacuta iterativ.
*/
// Varianta recursiva:

int nrTurnuri(int n)
{
   if(n==1) return 1;
   return 2*nrTurnuri(n-1) + 1;
}

int nrMonede(int n)
{
   if(n==1) return 1;
   return 2*nrMonede(n-1) + n;
```

```
}

void turnuri_apelFRec(int n, int& nrT, int& nrM)
{
   nrT = nrTurnuri(n);
   nrM = nrMonede(n);
}

// Numarul de turnuri este dat de formula:
//nrTurnuri(n) = 1 + 2 + 2^2 + ... + 2^(n-1)
// Numarul de monede este dat de formula:
//nrMonede(n) = n + (n-1)*2 + (n-2)*2^2 + ... + 1*2^(n-1)

void turnuri(int n, int &nrTurnuri, int &nrMonede) {
	nrMonede = 0;
	nrTurnuri = 0;
	int putere = 1;
	for (int inaltime = n; inaltime >= 1; inaltime--){
		nrTurnuri = nrTurnuri + putere;                                    // numarul turnurilor creste cu
urmatoraea putere a lui 2
		nrMonede = nrMonede + putere * inaltime;       // in fiecare turn avem "inaltime" monede
		putere = putere * 2;
	}
}

// Se pot folosi si formulele pt. numar total de turnuri / monede
// Formula pt. nr. turnuri::
// nrTurnuri(n) = 1 + 2 + 2^2 + ... + 2^(n-1) = 2^n -1
// Formula pt. nr. monede:
// nrMonede(n) = n + (n-1)*2 + (n-2)*2^2 + ... + 1*2^(n-1) =
//  = (1 + 2 + 2^2 + ... + 2^(n-1)) + (1 + 2 + 2^2 + ... + 2^(n-2)) + ... + (1 + 2) + 1 =
//  = (2^n-1) + (2^(n-1)-1) + ... + (2^2-1) + (2-1) =
//  = (2^n + 2^(n-1) + ... + 2^2 + 2) - n =
//  = (2^n + 2^(n-1) + ... + 2^2 + 2 + 1) - (n + 1) =
//  = (2^(n+1) - 1) - (n + 1) =
//  = 2^(n+1) - n - 2

void turnuri_Formula(int n, int &nrTurnuri, int &nrMonede) {
	int putere = 1;
	for (int i = 1; i <= n; i++){
		putere = putere * 2;
	}
	nrTurnuri = putere - 1;
	nrMonede = putere*2 - n - 2;
}
//---------------------------------------------------------------------------
int main(){
	int n, nrTurnuri, nrMonede;
	cout << "Introduceti inaltimea celui mai inalt turn: ";
	cin >> n;
	turnuri(n, nrTurnuri, nrMonede);
	cout << "nr. turnuri: " << nrTurnuri << endl;
	cout << "nr. monede: " << nrMonede << endl;

	turnuri_apelFRec(n, nrTurnuri, nrMonede);
	cout << "nr. turnuri: " << nrTurnuri << endl;
	cout << "nr. monede: " << nrMonede << endl;

	turnuri_Formula(n, nrTurnuri, nrMonede);
	cout << "nr. turnuri: " << nrTurnuri << endl;
	cout << "nr. monede: " << nrMonede << endl;
```

```
        return 0;
}
```

Subiectele probelor scrise de informatică, admitere nivel licenţă, Iulie 2016, Subiectul I
Problema NUMERE MAGICE

```
#include <iostream>
using namespace std;

short* reprezentare_baza(short p, int n){
   short* cifre = new short[10] {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
   while (n > 0) {
      cifre[n % p] = 1;
      n = n / p;
   }
   return cifre;
}

bool este_magic_1(short p, short q, int n) {
   // vector de frecventa
   short* cifre_bazap = reprezentare_baza(p, n);
   short* cifre_bazaq = reprezentare_baza(q, n);

   bool ok = true;
   short  i = 0;
   while (ok && i<10) {
      if (cifre_bazap[i] != cifre_bazaq[i]) {
         ok = false;
      }
      i++;
   }
   delete[] cifre_bazap;
   delete[] cifre_bazaq;

   return ok;
}

bool este_magic_2(short p, short q, int n) {
   // vector de frecventa
   short* cifre_bazap = reprezentare_baza(p, n);
   bool ok = true;
   int copie_n = n;
   while (ok && copie_n > 0) {
      short rest = copie_n % q;
      copie_n = copie_n / q;
      if (cifre_bazap[rest] == 0) {
         ok = false;
      }
      cifre_bazap[rest]++; // crestem numarul de aparitii
   }

   for (int i = 0; i< 10; i++) {
      if (cifre_bazap[i] == 1) {
         ok = false;
         // pot fi cifre folosite in baza p, dar sa nu fie folosite
         // in baza q
      }
   }
   delete[] cifre_bazap;
   return ok;
}
```

6 Decembrie 2025; orele 12; Asist. Univ. Drd. Coste Claudia Ioana

```
bool verif_cifre_baze(short p, short q, int n) {
// verificam daca cifrele cu care il scriem pe n in baza p
// sunt folosite si in baza q
   int copie_n = n;
   while (copie_n > 0) {
      short rest = copie_n % p;
      copie_n = copie_n / p;

      // verificam daca cifra rest apare si in scrierea numarului in baza q
      int copie_n2 = n;
      bool ok = false;
      while (!ok && copie_n2 > 0) {
         short rest2 = copie_n2 % q;
         copie_n2 = copie_n2 / q;
         if (rest == rest2) {
            ok = true;
         }
      }

      if (ok == false) {
         return false;
      }
   }
   return true;
}

bool este_magic_3(short p, short q, int n) {
   // verificam dubla incluziune
   // true daca cifrele cu care e scris n in baza p sunt printre cifrele cu care scriem n in baza q si daca daca cifrele cu care e scris n
in baza q sunt printre cifrele cu care scriem n in baza p
   return verif_cifre_baze(p, q, n) && verif_cifre_baze(q, p, n);
}

void sirNrMagice(short p, short q, int n, int &k, int sir[]){
   for (int i = 0; i<n; i++) {
      if (este_magic_2(p, q, i) == true) {
         k++;
         sir[k] = i;
         // sir[k++] = i;
      }
   }
   return ;
}

void tiparesteSir(int k, int sir[]){
   for (int i = 1; i<= k; i++) {
      cout<< sir[i]<<' ';
   }
   cout << endl;
   return ;
}

int main() {
   //cout << este_magic_1(7, 9, 31) <<endl; /*este magic*/
   //cout << este_magic_1(3, 9, 9) <<endl; /* este magic*/
   //cout << este_magic_1(7, 5, 31) <<endl; /*nu este*/
   //cout << este_magic_1(3, 7, 9) <<endl; /*nu este*/

   // cout << este_magic_2(7, 9, 31) <<endl; /*este magic*/
   // cout << este_magic_2(3, 9, 9) <<endl; /* este magic*/
   // cout << este_magic_2(7, 5, 31) <<endl; /*nu este*/
   // cout << este_magic_2(3, 7, 9) <<endl; /*nu este*/
```

```
   // cout << este_magic_3(7, 9, 31) <<endl; /*este magic*/
   // cout << este_magic_3(3, 9, 9) <<endl; /* este magic*/
   // cout << este_magic_3(7, 5, 31) <<endl; /*nu este*/
   // cout << este_magic_3(3, 7, 9) <<endl; /*nu este*/

            int n;
            cout << "Introduceti numarul n: ";
            cin >> n;
            short p;
            cout << "Introduceti baza p: ";
            cin >> p;
            short q;
            cout << "Introduceti baza q: ";
            cin >> q;
            int sir[10000];
            int k = 0;
            sirNrMagice(p, q, n, k, sir);
            tiparesteSir(k, sir);
   return 0;
}
```

Subiectele probelor scrise de informatică, admitere nivel licenţă, septembrie 2015, Problema a), b), subiectul I

A

```
#include <iostream>
using namespace std;

int suma_cifrelor(long a) {
   int s = 0;
   while (a > 0) {
      s = s+ a%10;
      a=a/10;
   }
   return s;
}

bool este_deosebit(long n){
   int i;
   for (i=n-1; i>0; i--) {
      if (i+suma_cifrelor(i) == n){
         return true;
      }
   }
   return false;
}

int main() {
   cout << este_deosebit(15) << endl;
   return 0;
}
```

B.

```
#include <iostream>
#include <cmath>
using namespace std;

bool descompunere_factori(long n){
   long root = sqrt(n);
   for (int i = 2; i < root; i++) {
      int counter = 0;
      while (n % i == 0) {
         n = n/i;
         counter++;
      }
      if (counter % 2 == 0) {
         return true;
      }
   }
   return false;
}

int main() {
   cout << descompunere_factori(12) << endl;
   cout << descompunere_factori(6) << endl;
   cout << descompunere_factori(36) << endl;
   cout << descompunere_factori(49) << endl;
   return 0;
}
```