

# Portofoliu Algoritmi și complexitate - Probleme rezonabile

Anul I, semestrul 2

Numele tău aici

Contribuitori:

- Sabina
- Carla

- Dacă vreți probleme rezolvate mai grele intrați aici;
- Dacă nu înțelegeți ceva întrebați-mă;
- Vă invit să mai scrieți și problemele voastre (da, știu);
- Nu garantez că problemele puse de alții sunt corecte (să citiți și cerințele);
- Dacă găsiți ceva greșit spuneți-mi;
- O parte din comentarii sunt pentru amuzamentul meu, iar în restul încerc să explic ce face (și de ce face) programul;
- Voi mai actualiza acest pdf când mai primim fișe de laboratoare sau când mai primesc exercitii rezolvate de la cineva. Deci intrați din nou pe același link (aici).

z sercem, Paбло.

## Laborator 1

8. Aproximați cu precizia  $\varepsilon$  limita la  $+\infty$  a șirului  $(s_n)_{n \in \mathbb{N}^*}$ , definit prin

$$x_n = \left(1 + \frac{1}{n}\right)^n, n \geq 1.$$

```
1 #include <iostream>
2 //includem cmath pentru ca avem nevoie de 'pow'
3 #include <cmath>
4
5 // 'using namespace std;' nu e cea mai bună practică,
6 // în special pentru proiecte mai mari.
7 // De exemplu, dacă definim o funcția asta:
8 //double riemann_zeta(double n) {
9 //    std::cout << "majie";
10 //    return 0;
11 //}
12 // și în main avem asta:
13 //cout << riemann_zeta(-1) << end;
14 // Când rulăm programul, pe ecran se afiseaza "majie0", are sens nu?
15 // Asta se intampla dacă compilăm cu versiuni mai vechi de c++17
16 // (o data cam la 3 ani mai apare o versiune de c++).
17 // Dar dacă compilăm cu c++17 (sau mai recent), pe ecran apare
18 // "-0.0833333". De ce?
19 // Păi din c++17 în "cmath" mai este definită și funcția "riemann_zeta",
20 // care sigur nu e definită ca funcția noastră (deși tot 'majie' e și
21 // acolo).
22 // Iar dacă nu știți asta și încercați ceva de genul ăsta, rămaneți
23 // fără păr în cap (sau fără tastatură) până aflați de ce nu va merge
24 // programul cum trebuie.
25 // Dar pentru ce facem noi la info, să zicem că e ok să scriem asta:
26 using namespace std;
27
28 // E (foarte) posibil ca aceste comentarii să fie foarte evidente dar
29 // nu știu altceva ce să scriu.
30 int main() {
31     // x0 ar fi  $x_{n-1}$  și x1,  $x_n$ 
32     double eps, x0 = 0, x1 = 0;
33     // delta e diferența între termenii consecutivi
34     // o initializăm cu un număr mare care sigur e mai mare decât eps
35     double delta = 10000;
36     cin << eps;
37
38     int n = 1;
39     // dacă ne simțeam aventuroși puteam scrie
40     // for (int n = 1; delta > eps; n++) {
41     while (delta > eps) {
42         // am incrementat n, deci  $x_{n-1}$  devine  $x_n$ 
43         x0 = x1;
44         // implementăm definiția care ni se da
```

```

45         x1 = pow(1 + 1.0 / n, n);
46         n++;
47         // actualiazm delta
48         delta = x1 - x0;
49     }
50     cout << x1 << endl;
51
52     return 0;
53 }

```

## 1. (Carla) BMI

```

1  #include <iostream>
2  using namespace std;
3
4  // Pablo: good luck
5  int main() {
6      unsigned kg, m;
7      float ma = 0;
8      cin >> kg >> m;
9      ma=kg/m^2;
10     if (ma<18.5)
11         cout << "risc minim pentru sanatate" << endl;
12     else if (ma>=18.5 && ma<25)
13         cout << "risc minim/scazut pentru sanatate" << endl;
14     else if (ma>=25 && ma<30)
15         cout << "risc scazut/moderat pentru sanatate" << endl;
16     else if (ma>=30 && ma<35)
17         cout << "risc moderat/ridicat pentru sanatate" << endl;
18     else cout << "risc ridicat pentru sanatate" << endl;
19     return 0;
20 }

```

## 2. (Carla) functia

```

1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4
5  int main() {
6      float x, f=0;
7      cin >> x;
8      if (x>=-5 && x<=5)
9          f=sqrt(abs(x))+6;
10     else f=5*x*x+7;
11     cout << f << endl;
12     return 0;
13 }

```

### 3. (Carla) alta functie

```
1 int functie(int x, int y, int a, int b) {
2     int c=0;
3     if (abs(x)<=8 && abs(y)<=8)
4         c=a*b*sqrt(x*x+y*y)*sin(x*x+y*y);
5     return c;
6 }
```

### 4. (Carla) cifra destinului (uau)

```
1 #include <iostream>
2 #include <cmath>
3 using namespace std;
4
5 int nastere(int zi, int luna, int an) {
6     int x = 0;
7     while (zi) {
8         x=x+zi%10;
9         zi=zi/10;
10    }
11    while (luna) {
12        x=x+luna%10;
13        luna=luna/10;
14    }
15    while (an) {
16        x=x+an%10;
17        an=an/10;
18    }
19    return x;
20 }
21
22 int destin(int x) {
23     unsigned s=0;
24     if (x<10) return x;
25     else while (x) {
26         s=s+x%10;
27         x=x/10;
28     }
29     return s;
30 }
31 int main() {
32     int a,b,c;
33     cin >> a >> b >> c;
34     int x=nastere(a,b,c);
35     cout << destin(x) << endl;
36     return 0;
37 }
```

5. (Carla) ăla cu multe subpuncte

a)

```
1 #include <iostream>
2 #include <cmath>
3 using namespace std;
4
5 int main () {
6     float a, b, c, d, x1, x2;
7     cin >> a >> b >> c; // coeficientii
8     d=b*b-4*a*c; //delta
9     x1=(-b-sqrt(d))/2*a; //prima solutie
10    x2=(-b+sqrt(d))/2*a; // a doua solutie
11    cout << x1 << ' ' << x2 << endl;
12    return 0;
13 }
```

b)

```
1 #include <iostream>
2 #include <cmath>
3 using namespace std;
4
5 int main() {
6     float x, n, max=-100000, min=100000;
7     cin >> n; //numarul de elemente
8     while (n) {
9         cin >> x; // element cu element
10        if (max<x)
11            max=x;
12        if (min>x)
13            min=x;
14        n--;
15    }
16    cout << max << ' ' << min << endl;
17    return 0;
18 }
```

k)

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     unsigned n,d;
6     cin>>n;
7     for(d=2;d<=n/2;d++)
8         if(n%d==0)
9             cout<<d<<" ";
```

```

10     cout << endl;
11     return 0;
12 }

```

## 6. (Carla) nr perfect

```

1  #include <iostream>
2
3  using namespace std;
4
5
6  // Pablo: poate fi bool, avem și d-ăștea in c++
7  int perfect(int n) {
8      int d;
9      int s = 1;
10     if (n<=2) return 0;
11     else {
12         for (d=2; d*d<=n; d++)
13             if (n%d==0) {
14                 s=s+d;
15                 if (n/d!=d)
16                     s=s+(n/d);
17             }
18         if (n==s)
19             return 1;
20     }
21     return 0;
22 }
23 int main() {
24     int n;
25     cin >> n;
26     if (perfect(n))
27         cout << n << " este perfect" << endl;
28     else cout << n << " nu este perfect" << endl;
29
30     return 0;
31 }

```

## 10. (Carla) Operatii cu multimi

### Apartenenta

```

1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6      int n, ok=0, c;
7      cin >> n;
8      int v[n+1];

```

```

9     for (int i=1;i<=n;i++)
10         cin >> v[i];
11     cin >> c;
12     for (int i=1;i<=n;i++)
13         if (v[i]==c)
14             ok=1;
15     if (ok==1)
16         cout << "apartine" << endl;
17     else cout << "nu apartine" << endl;
18
19     return 0;
20 }

```

Reuninunea

```

1  #include<iostream>
2  using namespace std;
3
4  int main() {
5      int a[20], b[20], c[50], i, j, m, n, k, gasit;
6      cin>>n;
7      for (i=0; i< n; i++) {
8          cin>>a[i];
9      }
10     cin>>m;
11
12     for(i=0; i<=m-1; i++) {
13         cin>>b[i];
14     }
15
16     k=0;
17     for(i=0; i<=n-1; i++) {
18         gasit=0;
19         for(j=0; j<=m-1 && !gasit; j++)
20             if (a[i]==b[j])
21                 gasit=1;
22         if(!gasit)
23             c[k++]=a[i];
24     }
25
26     for (i=0; i<=m-1; i++)
27         cout << b[i] << " ";
28     for (i=0; i<k; i++)
29         cout<<c[i]<<" ";
30
31     return 0;
32 }

```

## Laborator 2

### 2. (Inspirat de Sabina) Suma cifrelor - corectitudinea

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     // sper ca intelegeți ce face codul asta,
6     // daca nu incercati sa dati un exemplu pe o foaie
7     // și efectuati pas cu pas etapele algoritmului
8     // (puteti citi și ce scrie mai jos, deși nu cred că ajută)
9     int n, s = 0;
10    cin << n;
11    int i = 0;
12    while (n > 0) {
13        // se putea și cu 's += n%10; n /= 10;'
14        s = s + n % 10;
15        n = n / 10;
16        i++;
17    }
18    cout << "suma = " << s << endl;
19    return 0;
20 }
```

#### I. Parțial corectitudinea

Considerăm aserțiunile de intrare și ieșire:

$$P_{in} = \left\{ n = \sum_{j=0}^k c_j 10^j; c_j \in \overline{0,9}, \forall j \in \overline{0,k}; c_k \neq 0 \right\},$$
$$P_{out} = \left\{ s = \sum_{j=0}^k c_j \right\}.$$

Alegem proprietatea:

$$I = \left\{ n = \sum_{j=0}^{k-i} c_{i+j} 10^j; s = \sum_{j=0}^{i-1} c_{i-1-j} \right\}.$$

La intrarea in buclă:

$$i = 0$$
$$n = \sum_{j=0}^k c_j 10^j$$

Deci propoziția  $I = \left\{ n = \sum_{j=0}^k c_j 10^j; s = \sum_{j=0}^{-1} c_{-1-j} = 0 \right\}$  este adevărată.

Arătăm că propoziția  $I$  este invariantă.

Presupunem  $I$  adevărată la începutul iterației și  $n \neq 0$ ; demonstrăm  $I$  adevărată la sfârșitul iterației.

$$n = \sum_{j=0}^{n-i} c_{i+j} 10^j; s = \sum_{j=0}^{i-1} c_{i-1-j}$$



10

$s = s + n \% 10;$

$$s = \left( \sum_{j=0}^{i-1} c_{i-1-j} \right) + c_i = \sum_{j=0}^i c_{i-1-j}$$

11

$n = n / 10;$

$$n = \left\lfloor \left( \sum_{j=0}^{k-i} c_{i+j} 10^j \right) / 10 \right\rfloor = \left\lfloor \sum_{j=0}^{k-i} c_{i+j} 10^{j-1} \right\rfloor = \left\lfloor \sum_{j=1}^{k-i} c_{i+j} 10^{j-1} \right\rfloor + \lfloor c_i 10^{-1} \rfloor$$

$$\text{Cum } 0 \leq c_i \leq 9 \implies 0 \leq c_i 10^{-1} \leq 0.9 \implies \lfloor c_i 10^{-1} \rfloor = 0.$$

$$\text{Deci } n = \left\lfloor \sum_{j=1}^{k-i} c_{i+j} 10^{j-1} \right\rfloor = \sum_{j=1}^{k-i} c_{i+j} 10^{j-1} = \sum_{j=0}^{k-i-1} c_{i+j+1} 10^j.$$

12

$i++;$

Scriem  $res$  și  $n$  în funcție de noul  $i$ . Deci  $i$  devine  $i - 1$ .

$$s = \sum_{j=0}^{i-1} c_{i-1-j}$$

$$n = \sum_{j=0}^{k-(i-1)-1} c_{i-1+j+1} 10^j = \sum_{j=0}^{k-i} c_{i+j} 10^j$$

Deci  $I$  adevărată și la sfârșitul iterației.

La ieșirea din buclă:

$$i = k + 1$$

$$n = \sum_{j=0}^{k-(k+1)} c_{k+1+j} 10^j = \sum_{j=0}^{-1} c_{k+1+j} 10^j = 0$$

$$s = \sum_{j=0}^{k+1-1} c_{k+1-1-j} = \sum_{j=0}^k c_{k-j}$$

$$\text{Deci } P_{out} = \left\{ s = \sum_{j=0}^k c_{k-j} \right\} \text{ adevărată.}$$

În concluzie algoritmului este parțial corect.

## II. Total corectitudinea

Considerăm funcția  $t : \mathbb{N} \rightarrow \mathbb{N}$ ,  $t(i) = k + 1 - i$ ;

$t(i+1) - t(i) = k + 1 - (i+1) - (k + 1 - i) = -1 < 0$ , deci  $t$  monoton strict descrescătoare.

$$t(i) = 0 \iff i = k + 1 \iff n = \sum_{j=0}^{-1} c_{k+1+j} 10^j = 0 \iff \text{condiția de ieșire din buclă.}$$

În concluzie algoritmului este total corect.

## Laborator 3

2. (Inspirat de Sabina) Descrieți un algoritm pentru calculul produsului scalar a doi vectori din  $\mathbb{R}^n$ ...

$$(1) \text{ } ps = \langle x, y \rangle = \sum_{i=1}^n x_i y_i$$

```
1  #include <iostream>
2  using namespace std;
3
4  // e important sa numerotați liniile aici
5  int main() {
6      int n, x[100], y[100];
7      // citim marimea vectorului
8      cin >> n;
9      // citim vectorul, chestie destul de standard
10     cout << "x = ";
11     for (int i = 0; i < n; ++i)
12         cin >> x[i];
13     cout << "y = ";
14     for (int i = 0; i < n; ++i)
15         cin >> y[i];
16
17     // in variabila asta memoram produsul scalar
18     int ps = 0;
19     // (vezi formula de sus)
20     // sa spunem ca linia asta e z (15, sau cat e ea)
21     for (int i = 0; i < n; ++i) // z+1
22         ps = ps + x[i] * y[i]; // z+2
23
24     // nici 'endl' nu e cea mai grozava chestie:
25     cout << ps << endl;
26     return 0;
27 }
```

operația	cost	nr repetari	cost total
$z+2$	3	n	$3n$

$$T(n) = 3n$$

## Laborator 4

4. (Inspirat de Sabina) Cele mai mici 2 elemente dintr-o secvența

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     int n, x[100];
6     cin >> n;
7     //
8     for (int i = 0; i < n; ++i)
9         cin >> x[i];
10
11     // nu mai este 1989, putem declara variabile si la mijlocul functiei
12     int m1, m2;
13     // presupunem ca utilizatorul e rezonabil (nu e) si spunem ca n >= 2
14
15     // in m1 memoram cel mai mic element si in m2 al 2-lea cel mai mic,
16     // deci:  $m1 \leq m2$ 
17
18     // initializam cel m1, m2 cu primele 2 elemente din secvența
19     m1 = x[0];
20     m2 = x[1];
21
22
23     // ne asiguram ca m1 si m2 sunt in ordinea asteapata
24     // (adica  $m1 \leq m2$ ), iar daca nu le interschimbăm
25     if (m2 < m1) {
26         int t = m1;
27         m1 = m2;
28         m2 = t; //in loc de t putem folosi x[0]
29     }
30     for (int i = 2; i < n; i++) {
31         // daca x[i] e mai mic decat m1, atunci sigur e mai mic si decat
32         // m2, asadar curentul m1 e al doilea cel mai mic nr din secvența
33         // (pana acum), deci il punem in m2
34         // deoarece x[i] este cel mai mic, il punem in m1
35
36         if (x[i] <= m1) { // linia z
37             m2 = m1;
38             m1 = x[i];
39         }
40         // aici am pus '<' si mai sus '<=' deoarece e posibil sa avem
41         // de doua ori cea mai mica valoare, si atunci m1 = m2 = min{x}
42         // daca ajungem sa comparăm x[i] cu m2, inseamna ca x[i] > m1
43         else if (x[i] < m2) { // linia λ
44             // daca ajungem aici, x[i] este al doilea cel mai mic
45             // element din secventa, si il punem in m2
46             m2 = x[i];
```

```

47     }
48 }
49 // afisam ce trebuie (primele 2 elemente)
50 cout << m1 << " " << m2 << endl;
51
52 // o alta metoda de rezolvare a problemei ar fi sa sortam secvența
53 // pana la al 2-lea element si sa afisam primele 2 elemente:
54 // (tot  $\Theta(n)$ )
55 //for (int i = 0; i < 2; i++)
56 //    for (int j = 0; j < n; j++)
57 //        if (x[j] < x[i]) {
58 //            int t = x[i]; x[i]= x[j]; x[i] =t;
59 //        }
60 // cout << x[0] << " " << x[1];
61
62 return 0;
63 }

```

Cazul cel mai favorabil ( $m1 \geq x[i] \forall i \in \{2, \dots, n-1\}$ ):  $T(n) = n-2$  (se execută doar linia ȳ).

Cazul cel mai defavorabil ( $m1 < x[i] \forall i \in \{2, \dots, n-1\}$ ):  $T(n) = 2n-4$  (se execută linia ȳ și ȳ).

Mereu  $T(n) \in \Theta(n)$ .

## Laborator 5

1. La o stație meteo ....

Notam cu  $p_i$  presiunea din ziua  $i$  si cu  $t_i$  temperatura din ziua  $i$ ,  $i \in \overline{1, n}$ .

Notam  $i \preceq j$  daca ziua  $i$  ar trebui sa apara inainte de ziua  $j$ :

$$i \preceq j \iff (t_i < t_j) \vee ((t_i = t_j) \wedge (p_i > p_j))$$

```
1  #include <iostream>
2  using namespace std;
3  struct Zi {
4      int temp;
5      int presiune;
6  };
7
8  int main() {
9      int n;
10     Zi x[100];
11
12
13     cin >> n;
14
15     //citim elementele tabloului
16     for (int i = 0; i < n; ++i)
17         cin >> x[i].temp >> x[i].presiune;
18
19     // daca vrei alt algoritm de sortare inlocuiți aici:
20     for (int i = 0; i < n; ++i) {
21         for (int j = i+1; j < n; ++j) {
22             // sortam cu relatia de ordine  $\preceq$  de mai sus
23             // (practic sortam cum spune cerința: mai intai crescator
24             // dupa temperatura, iar daca temperaturile sunt egale,
25             // descrescator dupa presiune)
26             if (x[i].temp > x[j].temp || (x[i].temp == x[j].temp &&
27                 x[i].presiune < x[j].presiune)) {
28                 Zi t = x[i];
29                 x[i] = x[j];
30                 x[j] = t;
31             }
32         }
33     }
34     // afisam vectorul dupa ce l-am sortat
35     for (int i = 0; i < n; ++i)
36         cout << "t:" << x[i].temp << ", p: " << x[i].presiune << endl;
37
38     return 0;
39 }
```

## Laborator 6

### 5. Ackermann

```
1 // Asta scrie in definitie, asta am scris.
2 int A(int m, int n) {
3     if (m == 0) {
4         return n + 1;
5     }
6     else {
7         if (n == 0) return A(m-1, 1);
8         else return A(m-1, A(m, n-1));
9     }
10 }
```

### 8. Baza 2

```
1 #include <iostream>
2 using namespace std;
3 // 47.145518,27.6036255 bdm tss
4
5 // in tabloul s stocam caracterele corespunzatoare cifrelor lui n in
6 // baza 2 in ordine inversa, daca n are z cifre in baza 2 atunci,
7 // s[0] = a (z-1)-a cifra, s[1] = a (z-2)-a cifra... s[z-1] - prima
8 // cifra, daca citim de la dreapta la stanga
9
10 // functia returneaza cate cifre are n in baza 2 (z de mai sus), si
11 // implicit, pana unde am modificat vectorul s
12 int baza2(int n, char s[], int i) {
13     if (n == 0) return i;
14     // se poate si mai scurt ( s[i] = '0' + n%2;)
15     if (n % 2 == 0)
16         s[i] = '0';
17     else s[i] = '1';
18     return baza2(n/2, s, i+1);
19 }
20
21 int main() {
22     char s[100];
23     int n;
24     cin >> n;
25
26     int len = baza2(n, s, 0);
27     // afisam cifrele in ordinea corecta
28     for (int i = len - 1; i >= 0; i--)
29         cout << s[i];
30     cout << endl;
31     return 0;
32 }
```

## Laborator 7

### 2. Fibonacci

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4
5
6  // pe scurt: "asta zice în curs"
7  double putere(double x, int n) {
8      if (n == 1) return x;
9      double r = putere(x, n / 2);
10     r = r * r;
11     if (n % 2 == 1) r = r * x;
12     return r;
13 }
14 // implementam formula data in cerinta
15 int fib(int n) {
16     // daca vreti sa mearga și fib(0)
17     // (momentan pt pow(x, 0), se blocheaza până dă stack overflow),
18     // scrieti si linia asta:
19     //if (n==0) return 0;
20
21     // puteti defini o variabila 'double sqrt5 = sqrt(5);'
22     return round(1.0 / sqrt(5) *
23                 (putere((1+sqrt(5))/2, n) - putere(1-sqrt(5)/2, n)));
24     // varianta mai eficienta
25     // return round(1.0 / sqrt(5) * (putere((1+sqrt(5))/2, n)));
26 }
27 int main() {
28     int n;
29     cin >> n;
30     cout << fib(n) << endl;
31     return 0;
32 }
```

5. (inspirat de Carla) exista  $v[x] == x$

```
1  #include <iostream>
2  using namespace std;
3
4  int cauta(int v[], int stanga, int dreapta) {
5      if (stanga == dreapta) {
6          if (v[stanga] == stanga) return stanga;
7          // am putea omite else-ul
8          else return -1;
9      }
10     int mijloc = (stanga+dreapta)/2;
11     if (v[mijloc] == mijloc)
12         return mijloc;
13     if (v[mijloc] < mijloc)
14         return cauta(v, mijloc, stanga);
15     else return cauta(v, dreapta, mijloc);
16
17 }
18 int main() {
19     int n, v[100];
20     cin << n;
21     for (int i = 0; i < n; i++)
22         cin << v[i];
23     cout << cauta(v, 0, n) << endl;
24     return 0;
25 }
```



## Laborator 8

### 6. Problema aia lunga cu integrala definita

```
1 // puteți sa nu scrieți liniile astea 2 si functia main
2 #include <iostream>
3 using namespace std;
4
5
6 //aici definim functia ce va fi integrata
7 double f(double x) { //f(x) = x^2
8     // mai schimbati si voi putin expresia asta, cum ar fi
9     // return x; return 1; return sin(x); ....
10    return x * x;
11 }
12
13 // Se poate face si ceva de genu asta ca f sa fie parametru:
14 //template<typename F>
15 //double integrate(F f, double a, double b, double eps = 1e-5) {
16 // sau așa:
17 //double integrate(double (*f)(double), double a, ...
18
19 // daca vreti sa va simtiti mai romani puteti scrie "integreaza"
20
21 // asta e formula data in cerinta
22 double integrate(double a, double b, double eps) {
23     double delta = b - a;
24     if (delta < eps) {
25         return delta * (f(a) + f(b)) / 2;
26     }
27     else {
28         double c = (a+b)/2;
29         return integrate(a, c, eps) + integrate(c, b, eps);
30     }
31 }
32 int main() {
33     double a, b, eps;
34     cin >> a >> b >> eps;
35     // cu varianta mai faină merge și ceva de genu ăsta:
36     // cout << integrate(sin, a, b) << endl;
37     // doar că n-ar trebui să stim d-astea
38
39     cout << integrate(a, b, eps) << endl;
40
41     return 0;
42 }
```

## Laborator 9

### 5. numere de 4 cifre cu suma cifrelor 11

```
1  #include <iostream>
2  using namespace std;
3
4  // blasfemie, x ar trebui sa fie parametru pt functii,
5  // dar asa ne spune in curs...
6  int x[4], n4;
7
8  // am putea declara si asa:
9  //constexpr int n = 4;
10 //int x[n];
11 // dar nici asta n-ar trebui sa stim, si tot blasfemie e
12
13 // "asa spune si in curs"
14 void afiseaza() {
15     for (int i = 0; i < n; i++)
16         cout << x[i];
17     cout << " ";
18 }
19 bool valid() {
20     int s = 0;
21     for (int i = 0; i < n; i++)
22         s += x[i];
23     return s == 11;
24 }
25
26 // "asa spune in curs"
27 void btr(int k) {
28     //int i = 0;
29     //if (k == 0) i = 1;
30     //for (; i < 10; i++) {...
31
32     //varianta mai hardcore ar fi:
33     //for (int i = k == 0; i < 10; i++) {...
34
35     for (int i = 0; i < 10; i++) {
36         // numerele nu incep cu 0
37         // sunt metode mult mai eficiente pt verificarea asta
38         // (vezi inceputul functiei)
39         if (k == 0 && i == 0) continue;
40
41         // putem verifica pe parcurs daca suma e mai mare decat 11,
42         // si daca e, trecem la urmatorul
43         // "exercitiu cititorului"
44         x[k] = i;
45         if (k == n - 1) {
46             // valid() == true e redundant
```

```

47         // 'valid()' deja e bool si 'valid() == true' e tot bool
48         // si nu face absolut nimic in plus
49         // if (valid()) e mai de bun simt
50         if (valid() == true)
51             afiseaza();
52     }
53     else {
54         btr(k+1);
55     }
56 }
57 }
58
59 int main() {
60     btr(0);
61     return 0;
62 }

```

## Laborator 10-11

### 5. Gidul turistic și drumetețiile sale

```
1 #include <iostream>
2 using namespace std;
3
4 struct Drumetie {
5     int nr, s;
6 };
7 // "asa ne spune in curs", dar f = s+n
8 // (final = start + nr de zile per calatorie)
9
10 // am putea pune codul de aici in 'planificare', dar banuiesc ca
11 // asa ati face majoritatea. (ceea ce nu e așa de rău)
12 void sortare(Drumetie v[], int m) {
13     // aici putem sa le sortam după ziua de incepere, pt ca ziua
14     // de final e mereu 's+n'
15     for (int i = 0; i < m; i++) {
16         for (int j = i+1; j < m; j++) {
17             if (v[j].s < v[i].s) {
18                 Drumetie t = v[i];
19                 v[i] = v[j];
20                 v[j] = t;
21             }
22         }
23     }
24 }
25
26 // funcția returneaza indicele ultimului spectacol, ceea ce e destul
27 // de idiot. (am putea returna k+1, adica lungimea)
28 int planificare(Drumetie a[], int m, int b[], int n) {
29     sortare(a, m);
30     b[0] = a[0].nr;
31     // in loc sa memoram acest u, am putea memora
32     // v[u].s+2
33     int u = 0;
34     int k = 0;
35     for (int i = 1; i < m; i++) {
36         if (a[i].s >= a[u].s+n) {
37             k += 1; // sau 'k++', ori și mai bine '++k'
38             b[k] = a[i].nr;
39             u = i;
40         }
41     }
42     return k;
43 }
44
45 int main() {
46     Drumetie v[100];
```

```

47     int m, n;
48     int b[100];
49
50     // ar fi indicat sa adaugati si ceva cum ar fi
51     //'cout << "introduce-ti m: ";....'
52     cin >> m >> n;
53     for (int i = 0; i < m; i++) {
54         int s;
55         cin >> s; // sau 'cin >> v[i].s;'
56         v[i].s = s;
57         v[i].nr = i+1;
58     }
59     int k = planificare(v, m, b, 2);
60
61     // daca returnam k+1, mergem până la k, ca oamenii normali
62     for (int i = 0; i < k+1; i++)
63         cout << b[i] << " ";
64     //int k = planificare();
65     return 0;
66 }

```

#### 9. interclasarea sirurilor (nu scrieti)

```

1 //nu scrieti
2 #include <iostream>
3 #include <vector>
4 #include <chrono>
5 #include <thread>
6
7 // ieseau liniile din pagina
8 using namespace std;
9 using namespace std::chrono;
10
11 // cel mai bun algoritm de sortare
12 // nu se poate mai bine de 0 comparatii
13 vector<unsigned> sleep_sort(const vector<vector<unsigned>>& x) {
14     // sleep_sort nu merge pt numere negative
15     // asa ca nu exista numere negative ;)
16     vector<unsigned> res;
17     std::vector<std::thread> threads;
18
19     for (auto& vec : x) {
20         for (auto& val : vec) {
21             threads.emplace_back([val, &res]() {
22                 std::this_thread::sleep_for(milliseconds(val));
23                 res.push_back(val);
24             });
25         }
26     }
27     for (auto& t : threads)

```

```

28         t.join();
29     return res;
30 }
31
32 using Clock = high_resolution_clock;
33 int main() {
34     std::vector<std::vector<unsigned>> x = {
35         { 1, 2, 3, 4, 6, 8, 10, 213, 100000 },
36         { 4, 7, 8, 12, 32 , 41, 57, },
37         { 1, 6, 7, 9, 14, 51},
38     };
39     auto t1 = Clock::now();
40     auto res = sleep_sort(x);
41     auto t2 = Clock::now();
42     // ne așteptam ca timpul de execuție să fie cea mai mare valoare
43     // din x adică doar 100000 ms (1 min 40 sec), în cazul nostru
44     // ceea ce e perfect rezonabil în vedere că avem 0 comparații
45     std::cout << "Timp de execuție: "
46               << duration_cast<milliseconds>(t2 - t1).count()
47               << " ms\n";
48
49     for (auto& a : res)
50         std::cout << a << ", ";
51     std::cout << "\n";
52     std::cout << "nr de comparații = 0\n";
53
54     return 0;
55 }

```

O rezolvare mai serioasă ar fi concatenarea subșirurilor și sortarea cu radix sort (sau alt algoritm care nu compară elementele între ele). Asta face rezolvarea noastră mai bună decât rezolvarea vrută de autoarea problemei (metoda greedy), care trebuie să compare cel puțin două elemente între ele.

## Laborator 12

### 1. Fibonacci

```
1 #include <iostream>
2 using namespace std;
3
4 int v[100];
5 //varianta descendenta
6 int F(int n) {
7     if (n == 0 || n == 1) return n;
8     if (v[n] == 0)
9         v[n] = F(n-1)+F(n-2);
10    return v[n];
11 }
12 //varianta ascendenta
13 int F(int n) {
14     // am putea pastra, doar ultimele 2 elemente ale sirului,
15     // nu tot sirul
16     v[0] = 0;
17     v[1] = 1;
18     for (int i = 2; i <= n; i++) {
19         v[i] = v[i-1]+v[i-2];
20     }
21     return v[n];
22 }
23
24 int main() {
25     int n;
26     cin >> n;
27     cout << F(n) << endl;
28     return 0;
29 }
```

### 2. Functia magica

```
1 #include <iostream>
2 using namespace std;
3
4 double v[100];
5 // a) varianta descendenta
6 double P(int n, double x) {
7     if (n == 0) return 1;
8     if (n == 1) return x;
9     if (v[n] == 0)
10         v[n] = ((2.0*n-1)/n) * x * P(n-1, x) + ((n-1.0)/n) * P(n-2, x);
11     return v[n];
12 }
13
14 // b) varianta iterativa
```

```

15 double P(int n, double x) {
16     if (n == 0) return 1;
17     if (n == 1) return x;
18     double p1 = 1, p2 = x;
19     for (int i = 2; i <= n; i++) {
20         double p = ((2.0*i-1) / i) * x * p2 + ((i-1.0) / i) * p1;
21         p1 = p2;
22         p2 = p;
23     }
24     return p2;
25 }
26
27 int main() {
28     int n;
29     double x;
30     // presupunem ca functia nu ar returna niciodata 0
31     // (desi ea poate) pentru ca simplifica putin lucrurile:
32     // ar trebui sa initializam vectorul v cu NAN (not a number)
33     // si sa verificam in P cu isnan in loc de 0
34
35     cin >> n >> x;
36     cout << P(n, x) << endl;
37     return 0;
38 }

```