



Ghid pentru testul cu vectori

22 probleme de pe pbinfo și 8 exerciții suplimentare. La fiecare: enunțul pe scurt, un exemplu, trei indicii, ideea pe pași, soluția comentată și întrebări de verificare (cu răspunsurile la final).

22 probleme pbinfo	8 exerciții în plus	13 luate cu 100p	9 de refăcut (4 cu E.C.)
------------------------------	-------------------------------	----------------------------	------------------------------------

De unde vin problemele și exemplele

- **Problemele marcate „pbinfo #...”** sunt exact cele 22 rezolvate pe pbinfo.ro (cu ID-ul fiecăreia și punctajul tău de acolo). Enunțurile din ghid sunt **rezumate**; textul complet e pe pbinfo.
- **Exemplele:** exemplele și testele din ghid sunt făcute pentru ghid (nu sunt copiate de pe pbinfo); exemplul oficial al fiecărei probleme e în enunțul de pe pbinfo.
- **Exercițiile marcate „exercițiu suplimentar”** nu sunt de pe pbinfo. Sunt făcute pentru antrenament, în același stil.
- Toate soluțiile sunt **compile și rulate automat** pe toate exemplele și testele din ghid, înainte de generarea PDF-ului.

Cum folosești ghidul

1. Citește bazele și cele 8 reguli. Scrie șablonul de pornire de 2–3 ori din memorie.
2. Învață funcțiile scurte (suma cifrelor, prim, cmmdc...). Apar în multe probleme.
3. La fiecare problemă: fă exemplul pe hârtie, citește indiciile pe rând, apoi acoperă soluția și încearcă s-o scrii singur(ă).
4. Începe cu problemele unde ai pierdut puncte.
5. Pe site poți scrie și rula codul direct în browser și îl poți verifica pe teste.

Cuprins

1. Bazele C# în 11 lecții scurte
2. 8 reguli care îți salvează puncte
3. Șablonul de pornire
4. Funcții de știut pe de rost
5. Cum citești un enunț și lista de verificat
6. Problemele, pe capitole:
 - 6.1 Baza: citire, min/max, afișare: Maxim, MinMax, stergere1, Afișare, SumaPare*, PozMax*
 - 6.2 Cifre, divizori, numere prime: Numarare5, Numarare4, EgaleOglindit, ElemPP, NrApPrime, Prime4, inlocuireK, SumaCifreMax*, Palindroame*
 - 6.3 Verificări (DA / NU): VerifDiferite, PareOrdonate, PalXXL, Identice, Proportionale, Crescator*
 - 6.4 Sortare: Sort10, KMici*
 - 6.5 Secvențe: Secvente, SumaInSecv, SecvMax*
 - 6.6 Interclasare (doi sau trei indici): Interclasare1, Interclasare3, comun, Interclasare*
7. Dicționar
8. Răspunsurile la întrebări

* exercițiu suplimentar (nu e de pe pbinfo)

1. Bazele C# în 11 lecții scurte

Pe site, fiecare lecție are și un exercițiu interactiv. Aici e rezumatul.

1. Cum arată un program C#

Un program e o rețetă: calculatorul execută instrucțiunile una după alta, de sus în jos. În C#, rețeta are mereu același «ambalaj». Odată ce-l recunoști, rămâne de înțeles doar partea din mijloc.

```
using System;

class Program
{
    static void Main()
    {
        Console.WriteLine("Salut!");
    }
}
```

- Totul se întâmplă în `Main`.
- Fiecare `{` are pereche un `}`.
- Fiecare instrucțiune se termină cu `;`.
- C# face diferența între litere mari și mici: `Console` ≠ `console`.

2. Variabile: cutii cu etichetă

O variabilă e o cutie cu un nume pe ea, în care ții o valoare. Tipul cutiei spune ce fel de lucruri încap în ea.

```
int n = 5;
long mare = 9000000000;
bool gasit = false;
string mesaj = "NU EXISTA";
```

- `int` pentru numere obișnuite, `long` pentru numere uriașe.
- `=` pune o valoare, `==` compară.
- `7 / 2 == 3` la `int`.

3. Citirea datelor: șablonul

La pbinfo, numerele pot veni pe un rând sau pe mai multe. Ca să nu-ți mai pese, citești tot textul o dată și îl tai în bucăți. Apoi iei bucățile pe rând, cu un «deget» (k).

```
char[] sep = { ' ', '\n', '\r', '\t' };
string[] t = Console.In.ReadToEnd().Split(sep, StringSplitOptions.RemoveEmptyEntries);
int k = 0;
int n = int.Parse(t[k++]);
int[] v = new int[n];
for (int i = 0; i < n; i++)
    v[i] = int.Parse(t[k++]);
```

- Copiază șablonul la fiecare problemă.
- Fiecare număr se ia cu `t[k++]`.
- Mărimea vectorului e `n`: `new int[n]`.

4. Afișarea: Write vs WriteLine

Evaluatorul de pe pbinfo compară răspunsul tău caracter cu caracter. Un rând nou în plus sau o literă mică în loc de mare înseamnă răspuns greșit, chiar dacă logica e perfectă.

```
Console.Write(1 + " ");
Console.Write(2 + " ");
Console.WriteLine();           // acum trecem la rand nou
Console.WriteLine("DA");
```

- «Pe același rând» → `Write`.
- «Pe rânduri separate» → `WriteLine`.
- Șterge afișările de test înainte să trimiți.

5. Decizii: if, else și operatorii

Programul ia decizii cu `if`: «dacă e adevărat, fă asta; altfel, fă altceva».

```
if (x % 2 == 0)
    Console.WriteLine("par");
else
    Console.WriteLine("impar");
```

- Par: `x % 2 == 0`. Impar: `x % 2 != 0`.
- Ultima cifră: `x % 10`. Fără ultima cifră: `x / 10`.
- `&&` = și, `||` = sau.

6. Bucle: for și while

O buclă repetă aceleași instrucțiuni. Cu ea parcurgi un vector, cifrele unui număr sau citești până la 0.

```
for (int i = 0; i < 5; i++)
    Console.Write(i + " ");    // 0 1 2 3 4
```

- Parcurgere vector: `for (int i = 0; i < n; i++)`.
- Invers: `for (int i = n - 1; i >= 0; i--)`.
- Perechi: `for i`, apoi `for (int j = i + 1; j < n; j++)`.

7. Vectori: un șir de cutii

Un vector e un rând de cutii lipite, fiecare cu numărul ei (indicele). În loc de 100 de variabile, ai una singură: `v`, iar cutiile sunt `v[0]`, `v[1]`, `v[2]` ...

```
int[] v = new int[4];    // v[0], v[1], v[2], v[3]
v[0] = 7;
v[3] = 2;
Console.WriteLine(v[0] + v[3]);    // 9
```

- Mărimea vectorului = `n`, nu un număr inventat.
- Indicii merg de la 0 la `n - 1`.
- Ultimul: `v[n - 1]`.

8. Cifrele unui număr

Două operații îți dau acces la orice cifră: `x % 10` îți arată ultima cifră, iar `x / 10` o taie. Le repeți cât timp mai sunt cifre.

```
while (x > 0)
{
    int cifra = x % 10;
    // ... fa ceva cu cifra ...
    x /= 10;
}
```

- Ultima cifră: `x % 10`.
- Taie ultima cifră: `x /= 10`.
- Bucla: `while (x > 0)`.

9. Funcții: mașinării mici

O funcție e o mașinărie cu un nume: bagi ceva în ea (parametrii) și scoate un rezultat (`return`). O scrii o dată și o folosești de câte ori vrei.

```
static bool EstePar(int x)
{
    return x % 2 == 0;
}

static void Main()
{
    if (EstePar(14))
        Console.WriteLine("DA");
}
```

- `return` dă rezultatul înapoi.
- Funcțiile stau în `class Program`, în afara lui `Main`.
- Funcțiile clasice: SumaCifre, Oglindit, EstePrim, Cmmdc, PatratPerfect.

10. Fișiere .in și .out

Unele probleme nu folosesc tastatura și ecranul, ci fișiere: citești din `nume.in` și scrii în `nume.out`. Restul programului e la fel.

```
string text = File.ReadAllText("minmax.in");  
// ... la fel ca de obicei ...  
StreamWriter fout = new StreamWriter("minmax.out");  
fout.WriteLine(mn + " " + mx);  
fout.Close();
```

- `using System.IO;`
- Numele exacte din enunț.
- `fout.Close();` la final.

11. Sortare cu Array.Sort

Nu trebuie să scrii tu un algoritm de sortare. C# îți dă `Array.Sort(v)`, care ordonează vectorul crescător.

```
Array.Sort(v); // crescator  
for (int i = n - 1; i >= 0; i--) // de la coada = descrescator  
    Console.Write(v[i] + " ");
```

- `Array.Sort(v)` = crescător.
- Descrescător = parcurgi de la `n - 1` la 0.

2. 8 reguli care îți salvează puncte

Aproape toate punctele pierdute au venit din greșeli mici, nu din logică.

01 Citește tot inputul pe bucăți inlocuireK: eroare

Nu te baza pe `ReadLine` rând cu rând. La `inlocuireK`, `k` și `n` erau pe același rând; la `Secvente`, numerele erau pe mai multe rânduri. Șablonul cu `ReadToEnd().Split(...)` merge mereu.

X AȘA NU

```
int k = int.Parse(Console.ReadLine()); //  
randul e "3 5" -> crapa  
int n = int.Parse(Console.ReadLine());
```

✓ AȘA DA

```
int K = int.Parse(t[k++]);  
int n = int.Parse(t[k++]);
```

02 Write vs WriteLine Sort10: -80p, ElemPP: -100p

`Write` lasă cursorul pe același rând, `WriteLine` trece la rând nou. Din cauza asta s-au pierdut puncte la `Sort10` (20) și `ElemPP` (0).

X AȘA NU

```
Console.WriteLine(v[i] + " "); // fiecare  
numar pe alt rand
```

✓ AȘA DA

```
Console.Write(v[i] + " "); // toate pe  
acelasi rand
```

03 Textul se scrie exact ca în enunț Maxim: -20p

„NU EXISTA” nu e același lucru cu „nu exista”. Evaluatorul compară literă cu literă. La `Maxim` s-au pierdut 20 de puncte din asta.

X AȘA NU

```
Console.WriteLine("nu exista");
```

✓ AȘA DA

```
Console.WriteLine("NU EXISTA");
```

04 Vectorul are mărimea n VerifDiferite: -60p

`new int[n]`, nu un număr ales la întâmplare. La `VerifDiferite`, `new int[200]` cu `n` până la 500 a dus la 40 de puncte.

X AȘA NU

```
int[] v = new int[200];
```

✓ AȘA DA

```
int[] v = new int[n];
```

05 Min / max pornesc de la primul element MinMax: trecut din noroc

`mn = mx = v[0]`. Nu porni de la 0, 10 sau 10000: dacă numerele sunt toate negative sau toate uriașe, răspunsul iese greșit.

X AȘA NU

```
int mn = 10000, mx = 10;
```

✓ AȘA DA

```
int mn = v[0], mx = v[0];
```

06 Șterge afișările de test Afisare: 0p

Orice `Console.WriteLine` pus «ca să văd ce se întâmplă» strică răspunsul. La Afisare a dus la 0 puncte.

X AȘA NU

```
for (int i = 0; i < n; i++)  
    Console.Write(v[i] + " "); // TEST -  
    uitat in cod!
```

✓ AȘA DA

```
// doar afisarile cerute de enunt
```

07 Fișiere: numele exact și Close() risc de 0p

Numele fișierelor se copiază din enunț (minmax.in / minmax.out). Fără `fout.Close()`, fișierul de ieșire poate rămâne gol.

X AȘA NU

```
StreamWriter fout = new  
StreamWriter("MinMax.txt");  
fout.WriteLine(rez); // si fara  
Close()
```

✓ AȘA DA

```
StreamWriter fout = new  
StreamWriter("minmax.out");  
fout.WriteLine(rez);  
fout.Close();
```

08 Compilează înainte să trimiți 4 probleme: 0p

4 din 22 de probleme au fost erori de compilare (E.C.): paranteze lipsă, `' '` vs `" "`, `=` în plus. Rulează programul o dată pe exemplul din enunț.

X AȘA NU

```
char[] x = { " " }; // " " e text, nu  
character  
int[] v = new int = [n]; // = in plus
```

✓ AȘA DA

```
char[] x = { ' ' };  
int[] v = new int[n];
```

3. Șablonul de pornire

Citește tot inputul o dată și îl taie în numere, deci nu mai contează dacă numerele sunt pe un rând sau pe mai multe. Variabila `k` arată la al câtelea număr ai ajuns: de fiecare dată când iei un număr, scrii `t[k++]`.

CÂND SE CITEȘTE DE LA TASTATURĂ

```
using System;

class Program
{
    static void Main()
    {
        char[] sep = { ' ', '\n', '\r', '\t' }; // spatiu, rand nou, tab
        string[] t = Console.In.ReadToEnd().Split(sep, StringSplitOptions.RemoveEmptyEntries);
        int k = 0; // "degetul": al catelea numar urmeaza
        int n = int.Parse(t[k++]);
        int[] v = new int[n];
        for (int i = 0; i < n; i++)
            v[i] = int.Parse(t[k++]);

        // ... rezolvarea ...

        Console.WriteLine(rezultat);
    }
}
```

CÂND SE CITEȘTE DIN FIȘIER (.IN / .OUT)

```
using System;
using System.IO;

class Program
{
    static void Main()
    {
        char[] sep = { ' ', '\n', '\r', '\t' };
        string text = File.ReadAllText("problema.in"); // numele exact din enunt
        string[] t = text.Split(sep, StringSplitOptions.RemoveEmptyEntries);
        int k = 0;
        int n = int.Parse(t[k++]);
        int[] v = new int[n];
        for (int i = 0; i < n; i++)
            v[i] = int.Parse(t[k++]);

        // ... rezolvarea ...

        StreamWriter fout = new StreamWriter("problema.out");
        fout.WriteLine(rezultat);
        fout.Close(); // OBLIGATORIU, altfel fisierul poate ramane gol
    }
}
```

Write sau WriteLine? `Write` afișează și rămâne pe rând; `WriteLine` afișează și trece la rândul următor. Când enunțul spune „separate prin spațiu”, folosești `Write(x + " ")`.

4. Funcții de știut pe de rost

Suma cifrelor

```
static int SumaCifre(int x)
{
    int s = 0;
    while (x > 0) { s += x % 10; x /= 10; }
    return s;
}
```

Aduni ultima cifră, o tai, repeți până nu mai sunt cifre. Folosită la: Numarare5.

Oglinditul

```
static long Oglindit(long x)
{
    long r = 0;
    while (x > 0) { r = r * 10 + x % 10; x /= 10; }
    return r;
}
```

Fiecare cifră nouă se pune la dreapta, iar cele vechi se «împing» spre stânga ($\cdot 10$). Folosită la: EgaleOglindit.

Număr prim

```
static bool EstePrim(int x)
{
    if (x < 2) return false;
    for (int d = 2; d * d <= x; d++)
        if (x % d == 0) return false;
    return true;
}
```

0 și 1 nu sunt prime. Cauți divizori doar până la radical ($d \cdot d \leq x$). Folosită la: NrApPrime, Prime4.

Cmmdc (Euclid)

```
static int Cmmdc(int a, int b)
{
    while (b != 0) { int r = a % b; a = b; b = r; }
    return a;
}
```

Împarți și păstrezi restul, până restul devine 0. Ultimul împărțitor e cmmdc-ul. Folosită la: Numarare4.

Pătrat perfect

```
static bool PatratPerfect(int x)
{
    int r = (int)Math.Sqrt(x);
    return r * r == x;
}
```

Iei radicalul, tai zecimalele și verifici dacă la pătrat dă înapoi numărul. Folosită la: ElemPP.

Sortare + descrescător

```
Array.Sort(v); // crescator
for (int i = n - 1; i >= 0; i--) // parcurs invers = descrescator
    Console.Write(v[i] + " ");
```

O singură linie sortează crescător. De la coadă obții ordinea descrescătoare. Folosită la: Sort10, Identice, Proportionale.

5. Cum citești un enunț

1. **Ce primesc?** Citește «Date de intrare»: câte numere, în ce ordine, pe ce rânduri. Scrie-le pe hârtie.
2. **De unde citesc?** Tastatură (Console) sau fișier (nume.in)? Asta schimbă doar începutul și finalul programului.
3. **Ce trebuie să afișez?** Citește «Date de ieșire»: un număr? DA/NU? mai multe numere pe un rând? Exact ce text?
4. **Fac exemplul de mână** la exemplul din enunț și rezolvă-l pe hârtie, fără cod. Dacă nu-l poți face de mână, nu-l poți programa.
5. **Cazurile speciale** Ce fac dacă nu există nicio soluție? Dacă $n = 1$? La egalitate pe cine aleg?

Lista de verificat înainte să trimiți

- ☐ Am folosit șablonul de citire (nu ReadLine).
- ☐ Vectorul e new int[n] (nu un număr fix).
- ☐ Tastatură sau fișiere? Am verificat în enunț.
- ☐ Numele fișierelor sunt copiate exact (dacă e cu fișiere).
- ☐ Am pus fout.Close() la final (dacă e cu fișiere).
- ☐ Write sau WriteLine, exact cum cere enunțul.
- ☐ Textele (DA, NU, NU EXISTA) sunt scrise exact ca în enunț.
- ☐ Am șters toate afișările de test.
- ☐ Am rulat programul pe exemplul din enunț și iese exact la fel.

6.1 Baza: citire, min/max, afișare

Citești numere, le parcurgi, găsești cel mai mic sau cel mai mare și afișezi corect.

De pe pbinfo: Maxim (#54), MinMax (#82), stergere1 (#1453), Afișare (#488). Exerciții suplimentare: SumaPare, PozMax.

Maxim

Cel mai mare număr dintr-un șir care se termină cu 0

Enunț rezumat. Textul complet: pbinfo.ro/probleme/54/maxim

Enunțul pe scurt

Primești numere unul după altul. Când apare **0**, te oprești. 0 e doar semnalul de «gata», nu face parte din numere.

Trebuie să afișezi **cel mai mare** număr citit. Dacă primul număr e chiar 0 (deci n-a existat niciun număr), afișezi exact **NU EXISTA**.

Comparație. E ca la un concurs de sărituri: concurenții vin pe rând, iar tu ții minte doar recordul de până acum. Când cineva sare mai sus, el devine noul record. Când se încheie concursul (apare 0), recordul e răspunsul.

PRIMEȘTI

Numere întregi, separate prin spații sau rânduri noi.
Ultimul număr e 0.

AFIȘEZI

Cel mai mare număr citit (fără 0-ul de final) sau
textul **NU EXISTA**.

Exemplu

INTRARE

4 9 2 7 0

IEȘIRE

9

De ce: Numerele sunt 4, 9, 2, 7. Cel mai mare e 9. Zeroul doar oprește citirea.

Exemplul de mai sus e făcut pentru ghid și verificat cu soluția; exemplul oficial e în enunțul de pe pbinfo.

Alte teste: 0 → **NU EXISTA** · -5 -2 -9 0 → -2 · 7 0 → 7

Indicii (citește-le pe rând)

1. Ai nevoie de o buclă care se repetă la nesfârșit (**while (true)**) și din care ieși cu **break** când numărul citit e 0.
2. Ține minte două lucruri: **max** (recordul) și **amCitit** (dacă ai văzut măcar un număr). Primul număr citit devine direct **max**.
3. La final: dacă **amCitit** a rămas **false**, scrii **NU EXISTA**, exact așa, cu litere mari și un spațiu. Altfel scrii **max**.

Ideea, pe pași

1. Citești tot inputul cu șablonul (în bucăți).
2. Iei numerele pe rând cu **t[k++]**.
3. Dacă numărul e 0, ieși din buclă cu **break**.
4. Dacă e primul număr SAU e mai mare decât **max**, îl pui în **max**.
5. După buclă: **NU EXISTA** dacă n-ai citit nimic, altfel afișezi **max**.

Soluția comentată (maxim.cs)

BUCATA 1 DIN 5

```
using System;

class Program
{
    static void Main()
    {
        // citim TOT inputul si il spargem in "bucati" (numere)
        char[] sep = { ' ', '\n', '\r', '\t' }; // separatori: spatiu, rand nou, tab
        string[] t = Console.In.ReadToEnd().Split(sep, StringSplitOptions.RemoveEmptyEntries);
        int k = 0;
```

- 1 Începutul din șablon. Tot ce se scrie la tastatură e citit o singură dată și tăiat în bucăți (vectorul `t`). `k` e un «deget» care arată la a câta bucată am ajuns.

BUCATA 2 DIN 5

```
int max = 0;
bool amCitit = false; // am citit macar un numar diferit de 0?
```

- 2 `max` va ține recordul. `amCitit` e un steguleț (`bool` = poate fi doar `true` sau `false`). Pornește `false` și devine `true` la primul număr adevărat.

BUCATA 3 DIN 5

```
while (true)
{
    int x = int.Parse(t[k++]);
    if (x == 0) break; // 0 opreste citirea
```

- 3 `while (true)` înseamnă «repetă la nesfârșit». `t[k++]` ia bucata curentă și mută degetul mai departe. `int.Parse` transformă textul "9" în numărul 9. Dacă e 0, `break` ne scoate din buclă.

BUCATA 4 DIN 5

```
if (!amCitit || x > max) // primul numar devine maxim direct
{
    max = x;
    amCitit = true;
}
```

- 4 `!amCitit` se citește «încă n-am citit nimic», iar `||` înseamnă «sau». Deci: dacă e primul număr SAU e mai mare decât recordul, devine noul record.

BUCATA 5 DIN 5

```
if (!amCitit)
    Console.WriteLine("NU EXISTA"); // ATENTIE: exact cu litere mari
else
    Console.WriteLine(max);
}
```

- 5 Afișăm răspunsul. Textul trebuie să fie identic cu cel din enunț, pentru că evaluatorul compară literă cu literă.

Ce a mers prost la încercarea ta: Ai scris "nu exista" cu litere mici. Evaluatorul compară textul exact, deci trebuia "NU EXISTA". Asta a costat 20 de puncte.

Atenție la:

- Textul de afișat se copiază exact din enunț (litere mari, spații).
- Nu porni cu `max = 0`: dacă toate numerele sunt negative, răspunsul ar ieși greșit.
- 0-ul de la final nu se ia în calcul.

Întrebări

1. Se citesc numerele `-3 -8 -1 0`. Ce afișează un program care pornește cu `max = 0` și nu folosește `amCitit`?

- A. -1
- B. 0
- C. -8
- D. NU EXISTA

2. Ce face `break` în interiorul lui `while (true)`?

- A. Oprește tot programul
- B. Iese din buclă și continuă cu ce e după ea
- C. Sare la următoarea repetare
- D. Afișează un rând nou

MinMax

Cel mai mic și cel mai mare element

Enunț rezumat. Textul complet: pbinfo.ro/probleme/82/minmax

Enunțul pe scurt

În fișierul `minmax.in` ai un număr `n` și apoi `n` numere.

Scrii în fișierul `minmax.out` cel mai mic și cel mai mare dintre ele, pe același rând, cu un spațiu între ele.

Comparație. Ca atunci când cauți cel mai ieftin și cel mai scump produs dintr-o listă de prețuri: primul preț e și «cel mai ieftin», și «cel mai scump» de până acum. Apoi compari fiecare preț următor cu ele.

PRIMEȘTI

Fișierul `minmax.in`: pe primul rând `n`, apoi `n` numere.

AFIȘEZI

Fișierul `minmax.out`: minimul și maximul, separate printr-un spațiu.

Exemplu

MINMAX.IN

```
5
12 4 30 7 4
```

MINMAX.OUT

```
4 30
```

De ce: Cel mai mic număr e 4 (apare de două ori, nu contează). Cel mai mare e 30.

Exemplul de mai sus e făcut pentru ghid și verificat cu soluția; exemplul oficial e în enunțul de pe pbinfo.

Alte teste: `1 / 8 → 8 8` · `4 / -3 -10 5 0 → -10 5`

Indicii (citește-le pe rând)

1. Pune numerele într-un vector, apoi pornește cu `mn = mx = v[0]`, adică primul element.
2. Parcurge de la `i = 1`: dacă `v[i] < mn`, actualizezi `mn`; dacă `v[i] > mx`, actualizezi `mx`.
3. Scrii cu `fout.WriteLine(mn + " " + mx);` și nu uita `fout.Close();` la final.

Ideea, pe pași

1. Citești din `minmax.in` cu șablonul pentru fișiere.
2. `mn = mx = v[0]` (primul element).
3. Parcurgi restul: actualizezi `mn` și `mx` când e cazul.
4. Scrii în `minmax.out` și închizi fișierul cu `Close()`.

Soluția comentată (minmax.cs)

BUCATA 1 DIN 4

```
using System;
using System.IO;

class Program
{
    static void Main()
    {
        // problema cu FISIERE: citim din minmax.in, scriem in minmax.out
        char[] sep = { ' ', '\n', '\r', '\t' }; // separatori: spatiu, rand nou, tab
        string text = File.ReadAllText("minmax.in");
        string[] t = text.Split(sep, StringSplitOptions.RemoveEmptyEntries);
        int k = 0;
```

- 1 Șablonul pentru fișiere. `File.ReadAllText("minmax.in")` citește tot fișierul ca text. Ca să meargă, ai nevoie de `using System.IO;` sus.

BUCATA 2 DIN 4

```
int n = int.Parse(t[k++]);
int[] v = new int[n];
for (int i = 0; i < n; i++)
    v[i] = int.Parse(t[k++]);
```

- 2 Citim n, facem un vector cu exact n locuri și îl umplem cu următoarele n numere.

BUCATA 3 DIN 4

```
// pornim de la PRIMUL element, nu de la valori inventate
int mn = v[0], mx = v[0];
for (int i = 1; i < n; i++)
{
    if (v[i] < mn) mn = v[i];
    if (v[i] > mx) mx = v[i];
}
```

- 3 Primul element e la început și minim, și maxim. Apoi îl comparăm pe fiecare element următor (de la `i = 1`) cu ele.

BUCATA 4 DIN 4

```
StreamWriter fout = new StreamWriter("minmax.out");
fout.WriteLine(mn + " " + mx);
fout.Close(); // FARA Close() fisierul poate ramane gol!
}
```

- 4 `StreamWriter` e «pixel» cu care scriem în fișier. `mn + " " + mx` lipește cele două numere cu un spațiu între ele. `Close()` salvează efectiv fișierul.

La încercarea ta: Ai luat 100, dar ai pornit cu `mx = 10` și `mn = 10000`. Numerele pot ajunge la 1.000.000, deci a trecut din noroc. Regula sigură: pornești mereu de la primul element.

Atenție la:

- Nu inventa valori de pornire (0, 10000...). Pornește de la `v[0]`.
- Numele fișierelor se scriu exact ca în enunț.
- Fără `fout.Close()` fișierul poate rămâne gol.

Întrebări

3. De ce e periculos să pornești cu `mn = 10000` ?

- A. Nu e periculos, merge mereu
- B. Dacă toate numerele sunt mai mari de 10000, minimul iese 10000, care nici nu e în listă
- C. Pentru că 10000 nu încapă în int
- D. Pentru că programul devine lent

4. Ce se poate întâmpla dacă uiți `fout.Close()` ?

- A. Nimic
- B. Eroare de compilare
- C. Fișierul de ieșire poate rămâne gol
- D. Se șterge fișierul .in

stergere1

Ștergi elementele pare din vector

Enunț rezumat. Textul complet: pbinfo.ro/probleme/1453/stergere1

Enunțul pe scurt

Primești un vector cu n numere.

Trebuie să «ștergi» toate numerele **pare** și să afișezi ce rămâne, în aceeași ordine.

Comparație. Ca la triat fructe de pe o bandă: le iei pe rând. Pe cele bune (impare) le așezi în față, unul lângă altul. Pe cele stricate (pare) nu le pui nicăieri. La final, în față ai doar fructele bune, în ordinea în care au venit.

PRIMEȘTI

n , apoi n numere.

AFIȘEZI

Numerele impare, în ordinea inițială, pe un rând.

Exemplu

INTRARE

7
3 8 5 2 10 9 4

IEȘIRE

3 5 9

De ce: Pare sunt 8, 2, 10, 4, deci le ștergem. Rămân 3, 5, 9.

Exemplul de mai sus e făcut pentru ghid și verificat cu soluția; exemplul oficial e în enunțul de pe pbinfo.

Alte teste: 4 / 2 4 6 1 → 1 · 3 / 1 3 5 → 1 3 5

Indicii (citește-le pe rând)

1. Nu trebuie să muți toate elementele spre stânga la fiecare ștergere. Există un truc mai simplu, cu un singur for.
2. Folosește un al doilea indice `m`, care arată unde pui următorul element «bun». Pornește de la 0.
3. Pentru fiecare `v[i]` impar: `v[m] = v[i]; m++;`. La final, noua lungime a vectorului e `m`.

Ideea, pe pași

1. Citești vectorul.
2. `m = 0` (câte elemente bune am păstrat până acum).
3. Pentru fiecare element impar: îl copiezi pe poziția `m` și crești `m`.
4. `n = m`, apoi afișezi primele n elemente.

Soluția comentată (stergere1.cs)

BUCATA 1 DIN 3

```
using System;

class Program
{
    static void Main()
    {
        char[] sep = { ' ', '\n', '\r', '\t' }; // separatori: spatiu, rand nou, tab
        string[] t = Console.In.ReadToEnd().Split(sep, StringSplitOptions.RemoveEmptyEntries);
        int k = 0;
        int n = int.Parse(t[k++]);
        int[] v = new int[n];
        for (int i = 0; i < n; i++)
            v[i] = int.Parse(t[k++]);
    }
}
```

- 1 Șablonul de citire: `n`, apoi vectorul cu `n` elemente.

BUCATA 2 DIN 3

```
// stergerea "pe bune": pastram in fata doar elementele impare
int m = 0; // noua lungime
for (int i = 0; i < n; i++)
    if (v[i] % 2 != 0)
    {
        v[m] = v[i];
        m++;
    }
n = m;
```

- 2 `v[i] % 2 != 0` înseamnă «restul împărțirii la 2 nu e 0», adică numărul e impar. Fiecare element impar e copiat pe poziția `m`, apoi `m` crește. Elementele pare sunt pur și simplu sărite.

BUCATA 3 DIN 3

```
for (int i = 0; i < n; i++)
    Console.Write(v[i] + " ");
Console.WriteLine();
}
```

- 3 Afișăm doar primele `n` (noua lungime) elemente, toate pe același rând (`Write`, nu `WriteLine`).

La încercarea ta: 100 de puncte. Tehnica cu `m` e cea mai simplă metodă de ștergere și merită învățată pe de rost.

Atenție la:

- Par = `x % 2 == 0`. Impar = `x % 2 != 0` (merge și pentru numere negative).
- După ștergere afișezi doar primele `m` elemente, nu toate `n`.

Întrebări

5. Vectorul `5 6 7`. După bucla de ștergere a numerelor pare, cât e `m`?

- A. 1
- B. 2
- C. 3
- D. 0

6. Ce înseamnă `v[i] % 2 != 0`?

- A. `v[i]` e par
- B. `v[i]` e impar
- C. `v[i]` e 0
- D. `v[i]` e negativ

Afisare

Pozițiile pare înainte, cele impare de la coadă

Enunț rezumat. Textul complet: pbinfo.ro/probleme/488/afisare

Enunțul pe scurt

Primești n numere. Pozițiile se numără de la 1 (primul element e pe poziția 1).

Pe **primul rând** afișezi elementele de pe pozițiile pare: 2, 4, 6, ... (în ordine).

Pe **al doilea rând** afișezi elementele de pe pozițiile impare, de la coadă spre început: cea mai mare poziție impară, apoi cu 2 mai puțin, ... până la 1.

Comparație. Ai o coadă de 7 oameni numerotați de la 1 la 7. Întâi îi strigi pe cei cu număr par, de la începutul cozii (2, 4, 6). Apoi, pe alt rând, îi strigi pe cei cu număr impar, pornind de la capătul cozii (7, 5, 3, 1).

PRIMEȘTI

n , apoi n numere.

AFIȘEZI

Două rânduri: elementele de pe poziții pare (crescător), apoi cele de pe poziții impare (descrescător).

Exemplu

INTRARE

7
10 20 30 40 50 60 70

IEȘIRE

20 40 60
70 50 30 10

De ce: Pozițiile pare 2, 4, 6 au valorile 20, 40, 60. Pozițiile impare de la coadă, 7, 5, 3, 1, au valorile 70, 50, 30, 10.

Exemplul de mai sus e făcut pentru ghid și verificat cu soluția; exemplul oficial e în enunțul de pe pbinfo.

Alte teste: 6 / 1 2 3 4 5 6 → 2 4 6 / 5 3 1

Indicii (citește-le pe rând)

- Ca să nu te încurci, citește vectorul cu indici de la 1, exact ca în enunț: `int[] v = new int[n + 1];` și `for (int i = 1; i <= n; i++)`.
- Rândul 1: `for (int i = 2; i <= n; i += 2)`. Între cele două rânduri pui `Console.WriteLine();`.
- Rândul 2: pornești de la cel mai mare număr impar $\leq n$ (dacă n e impar e chiar n , altfel $n - 1$) și scazi câte 2 până la 1.

Ideea, pe pași

- Citești vectorul în `v[1..n]`.
- Afișezi `v[2]`, `v[4]`, ... pe primul rând.
- Treci la rând nou.

4. Afișezi `v[start]`, `v[start-2]`, ..., `v[1]`, unde `start` e cel mai mare impar $\leq n$.

Soluția comentată (afisare.cs)

BUCATA 1 DIN 4

```
using System;

class Program
{
    static void Main()
    {
        char[] sep = { ' ', '\n', '\r', '\t' }; // separatori: spatiu, rand nou, tab
        string[] t = Console.In.ReadToEnd().Split(sep, StringSplitOptions.RemoveEmptyEntries);
        int k = 0;
        int n = int.Parse(t[k++]);
```

1 Șablonul de citire, până la `n`.

BUCATA 2 DIN 4

```
int[] v = new int[n + 1]; // folosim indici de la 1 la n
for (int i = 1; i <= n; i++)
    v[i] = int.Parse(t[k++]);
```

2 Facem vectorul cu un loc în plus (`n + 1`) și nu folosim `v[0]`. Așa, poziția 1 din enunț e chiar `v[1]` și nu mai trebuie să faci socoteli cu ± 1 .

BUCATA 3 DIN 4

```
// linia 1: indici pari, crescator: 2, 4, 6, ...
for (int i = 2; i <= n; i += 2)
    Console.Write(v[i] + " ");
Console.WriteLine();
```

3 `i += 2` înseamnă că sărim din 2 în 2: 2, 4, 6... `Console.WriteLine()` fără nimic în paranteze doar trece la rândul următor.

BUCATA 4 DIN 4

```
// linia 2: indici impari, descrescator: pornim de la cel mai mare impar <= n
int start = (n % 2 == 1) ? n : n - 1;
for (int i = start; i >= 1; i -= 2)
    Console.Write(v[i] + " ");
Console.WriteLine();
}
```

4 `(conditie) ? A : B` e un «if pe scurt»: dacă `n` e impar, `start = n`, altfel `start = n - 1`. Apoi coborâm din 2 în 2 până la 1.

Ce a mers prost la încercarea ta: Ai lăsat în cod o afișare de test (toate elementele), ai amestecat indicii de la 0 cu cei de la 1 și n-ai trecut la rând nou între cele două linii. Rezultat: 0 puncte, deși ideea era aproape.

Atenție la:

- Șterge orice afișare pusă «ca să văd ce se întâmplă».
- Hotărăște-te: indici de la 0 SAU de la 1, nu amândouă.
- Între cele două rânduri: `Console.WriteLine();`.

Întrebări

7. Pentru $n = 8$, de la ce poziție pornește al doilea rând?

- A. 8
- B. 7
- C. 1
- D. 9

8. Ce afișează `for (int i = 2; i <= 7; i += 2) Console.Write(i + " ");`?

- A. 2 3 4 5 6 7
- B. 2 4 6
- C. 2 4 6 8
- D. 1 3 5 7

SumaPare

Suma numerelor pare din vector

Făcut pentru antrenament, în stilul problemelor de pe pbinfo. Soluția e verificată pe toate testele.

Enunțul pe scurt

Primești **n** numere întregi.

Afișezi **suma** celor care sunt **pare**. Dacă nu există niciun număr par, suma e 0.

Comparație. Ca atunci când aduni doar bancnotele de un anumit fel dintr-un teanc: le iei pe rând și le pui deoparte doar pe cele care se potrivesc.

PRIMEȘTI

n, apoi n numere întregi.

AFIȘEZI

Suma numerelor pare.

Exemplu

INTRARE

6
3 8 5 2 10 7

IEȘIRE

20

De ce: Pare sunt 8, 2 și 10. $8 + 2 + 10 = 20$.

Alte teste: 3 / 1 3 5 → 0 · 4 / -2 4 -6 1 → -4

Indicii (citește-le pe rând)

1. Pornești cu o variabilă `suma = 0` și o crești pe parcurs.
2. Un număr e par dacă `x % 2 == 0` (merge și pentru negative).
3. În for: dacă `v[i]` e par, faci `suma += v[i];`. La final afișezi `suma`.

Ideea, pe pași

1. Citești vectorul.
2. `suma = 0`.
3. Pentru fiecare element par: `suma += v[i]`.
4. Afișezi suma.

Soluția comentată (sumapare.cs)

BUCATA 1 DIN 3

```
using System;

class Program
{
    static void Main()
    {
        char[] sep = { ' ', '\n', '\r', '\t' }; // separatori: spatiu, rand nou, tab
        string[] t = Console.In.ReadToEnd().Split(sep, StringSplitOptions.RemoveEmptyEntries);
        int k = 0;
        int n = int.Parse(t[k++]);
        int[] v = new int[n];
        for (int i = 0; i < n; i++)
            v[i] = int.Parse(t[k++]);
    }
}
```

1 Citirea din șablon.

BUCATA 2 DIN 3

```
long suma = 0; // suma poate iesi mare -> long
for (int i = 0; i < n; i++)
    if (v[i] % 2 == 0)
        suma += v[i];
```

2 `suma += v[i]` e prescurtarea pentru `suma = suma + v[i]`. Folosim `long` pentru că multe numere mari adunate pot depăși `int`.

BUCATA 3 DIN 3

```
Console.WriteLine(suma);
}
}
```

3 Afișăm rezultatul.

Atenție la:

- Suma pornește de la 0, nu de la `v[0]`.
- Testul de par e `% 2 == 0`, nu `/ 2 == 0`.

Întrebări

9. Ce face `suma += v[i];` ?

- A. Pune `v[i]` în suma
- B. Adună `v[i]` la suma
- C. Compară suma cu `v[i]`
- D. Scade `v[i]`

10. Pentru `1 2 3 4`, cât e suma numerelor pare?

- A. 10
- B. 4
- C. 6
- D. 2

PozMax

Poziția primei apariții a maximului

Făcut pentru antrenament, în stilul problemelor de pe pbinfo. Soluția e verificată pe toate testele.

Enunțul pe scurt

Primești n numere întregi.

Afișezi **poziția** (numerotată de la **1**) pe care apare **cel mai mare** număr. Dacă maximul apare de mai multe ori, afișezi **prima** poziție.

Comparație. Într-un clasament, nu te interesează doar nota cea mai mare, ci și cine a luat-o primul: ții minte locul, nu doar valoarea.

PRIMEȘTI

n , apoi n numere întregi.

AFIȘEZI

Poziția (de la 1) a primului maxim.

Exemplu

INTRARE

6
4 9 2 9 1 7

IEȘIRE

2

De ce: Maximul e 9. Apare pe pozițiile 2 și 4; o afișăm pe prima, 2.

Alte teste: 1 / 5 → 1 · 4 / -3 -1 -1 -8 → 2

Indicii (citește-le pe rând)

1. Nu ține minte valoarea maximului, ci **indicele** lui: `poz = 0` la început.
2. Pentru fiecare `i` de la 1: dacă `v[i] > v[poz]`, atunci `poz = i`.
3. Folosește `>` strict, ca la egalitate să rămână prima poziție. La final afișezi `poz + 1`.

Ideea, pe pași

1. Citești vectorul.
2. `poz = 0`.
3. Dacă găsești un element mai mare decât `v[poz]`, `poz = i`.
4. Afișezi `poz + 1`.

Soluția comentată (pozmax.cs)

BUCATA 1 DIN 3

```
using System;

class Program
{
    static void Main()
    {
        char[] sep = { ' ', '\n', '\r', '\t' }; // separatori: spatiu, rand nou, tab
        string[] t = Console.In.ReadToEnd().Split(sep, StringSplitOptions.RemoveEmptyEntries);
        int k = 0;
        int n = int.Parse(t[k++]);
        int[] v = new int[n];
        for (int i = 0; i < n; i++)
            v[i] = int.Parse(t[k++]);
    }
}
```

1 Citirea din șablon.

BUCATA 2 DIN 3

```
int poz = 0; // indicele maximului (de la 0)
for (int i = 1; i < n; i++)
    if (v[i] > v[poz]) // > strict: la egalitate ramane primul
        poz = i;
```

2 Ținem minte unde e maximul, nu cât e. Valoarea o putem afla oricând: `v[poz]`.

BUCATA 3 DIN 3

```
Console.WriteLine(poz + 1); // pozitiile se numara de la 1
}
}
```

3 În C# indicii pornesc de la 0, dar enunțul numără de la 1, deci adunăm 1.

Atenție la:

- > strict pentru prima apariție (>= ar da ultima).
- Afișezi `poz + 1`, nu `poz`.

Întrebări

11. Vectorul `5 8 8 3`. Ce afișează programul dacă folosește `>=` în loc de `>`?

- A. 2
- B. 3
- C. 8
- D. 1

12. De ce se afișează `poz + 1`?

- A. Din greșeală
- B. Indicii C# pornesc de la 0, enunțul numără de la 1
- C. Ca să sară peste maxim
- D. Pentru că n începe de la 1

6.2 Cifre, divizori, numere prime

Lucrezi cu cifrele unui număr ($\% 10$ și $/ 10$), cu cmmdc și cu numere prime.

De pe pbinfo: Numarare5 (#499), Numarare4 (#496), EgaleOglindit (#294), ElemPP (#132), NrApPrime (#100), Prime4 (#4456), inlocuireK (#383). Exerciții suplimentare: SumaCifreMax, Palindroame.

Numarare5

Perechi de numere cu aceeași sumă a cifrelor

Enunț rezumat. Textul complet: pbinfo.ro/probleme/499/numarare5

Enunțul pe scurt

Primești n numere.

Numeri câte **perechi** de elemente au aceeași **sumă a cifrelor**. O pereche înseamnă două elemente de pe poziții diferite. Perechea (A, B) e aceeași cu (B, A), deci o numeri o singură dată.

Comparație. La o petrecere, fiecare om dă mâna cu fiecare alt om o singură dată. Tu numeri doar strângerile de mână dintre oameni care au același «număr norocos» (suma cifrelor).

PRIMEȘTI

n , apoi n numere naturale.

AFIȘEZI

Numărul de perechi cu aceeași sumă a cifrelor.

Exemplu

INTRARE

5
12 30 5 21 14

IEȘIRE

4

De ce: Sumele cifrelor: $12 \rightarrow 3$, $30 \rightarrow 3$, $5 \rightarrow 5$, $21 \rightarrow 3$, $14 \rightarrow 5$. Perechi cu suma 3: (12,30), (12,21), (30,21). Perechi cu suma 5: (5,14). Total: 4.

Exemplul de mai sus e făcut pentru ghid și verificat cu soluția; exemplul oficial e în enunțul de pe pbinfo.

Alte teste: 3 / 1 10 100 → 3 · 4 / 1 2 3 4 → 0

Indicii (citește-le pe rând)

1. Scrie întâi o funcție `SumaCifre(x)`: aduni ultima cifră (`x % 10`) și o tai (`x /= 10`), cât timp `x > 0`.
2. Ca să iei fiecare pereche o singură dată, folosești două for-uri: `i` de la 0, iar `j` de la `i + 1`.
3. Dacă `SumaCifre(v[i]) == SumaCifre(v[j])`, crești contorul: `cnt++`.

Ideea, pe pași

1. Funcția `SumaCifre`.
2. Citești vectorul.
3. Pentru fiecare pereche (i, j) cu $j > i$, compari sumele cifrelor.
4. Afișezi contorul.

Soluția comentată (numarare5.cs)

BUCATA 1 DIN 3

```
using System;

class Program
{
    static int SumaCifre(int x)
    {
        int s = 0;
        while (x > 0)
        {
            s += x % 10;    // ultima cifra
            x /= 10;        // o taiem
        }
        return s;
    }
}
```

- 1 O funcție e o «mașinărie»: primește un număr `x` și dă înapoi (`return`) suma cifrelor lui. `x % 10` e ultima cifră (din 123 → 3), iar `x /= 10` taie ultima cifră (123 → 12).

BUCATA 2 DIN 3

```
static void Main()
{
    char[] sep = { ' ', '\n', '\r', '\t' }; // separatori: spatiu, rand nou, tab
    string[] t = Console.In.ReadToEnd().Split(sep, StringSplitOptions.RemoveEmptyEntries);
    int k = 0;
    int n = int.Parse(t[k++]);
    int[] v = new int[n];
    for (int i = 0; i < n; i++)
        v[i] = int.Parse(t[k++]);
}
```

- 2 Citirea obișnuită din șablon.

BUCATA 3 DIN 3

```
int cnt = 0;
for (int i = 0; i < n; i++)
    // j > i => fiecare pereche o singura data
    for (int j = i + 1; j < n; j++)
        if (SumaCifre(v[i]) == SumaCifre(v[j]))
            cnt++;

Console.WriteLine(cnt);
}
```

- 3 Două for-uri unul în altul. Pentru că `j` pornește de la `i + 1`, perechea (0,1) e luată, dar (1,0) nu. Așa fiecare pereche e numărată o dată.

Ce a mers prost la încercarea ta: Eroare de compilare: ai scris `char[] x = {" "}`. Ghilimelele duble înseamnă text (string); pentru un singur caracter trebuia `' '`. Apoi ai apelat `cif(x[i])` pe vectorul de separatori, în loc de vectorul cu numere.

Atenție la:

- `' '` (apostrof) = un caracter. `" "` (ghilimele) = text.
- Al doilea for pornește de la `i + 1`, nu de la 0.

Întrebări

13. Câte `SumaCifre(907)` ?

- A. 16
- B. 97
- C. 7
- D. 9

14. Câte perechi (i, j) cu $i < j$ există într-un vector cu 4 elemente?

- A. 4
- B. 6
- C. 8
- D. 16

Numarare4

Câte numere sunt prime cu ultimul

Enunț rezumat. Textul complet: pbinfo.ro/probleme/496/numarare4

Enunțul pe scurt

Primești n numere.

Numeri câte dintre ele (fără ultimul) sunt **prime cu ultimul element**.

Atenție: «prime între ele» NU înseamnă că numerele sunt prime! Înseamnă că singurul divizor comun e 1, adică **cmmdc-ul lor e 1**. De exemplu 8 și 9 sunt prime între ele, deși niciunul nu e prim.

Comparație. Gândește-te la divizori ca la «prieteni comuni». Două numere sunt prime între ele când nu au niciun prieten comun în afară de 1, care e prietenul tuturor.

PRIMEȘTI

n , apoi n numere naturale nenule.

AFIȘEZI

Câte dintre primele $n-1$ numere au cmmdc 1 cu ultimul.

Exemplu

INTRARE

6
5 9 25 8 7 12

IEȘIRE

3

De ce: Ultimul e 12. $\text{cmmdc}(5,12)=1$ ✓, $\text{cmmdc}(9,12)=3$ ✗, $\text{cmmdc}(25,12)=1$ ✓, $\text{cmmdc}(8,12)=4$ ✗, $\text{cmmdc}(7,12)=1$ ✓. Trei numere.

Exemplul de mai sus e făcut pentru ghid și verificat cu soluția; exemplul oficial e în enunțul de pe pbinfo.

Alte teste: 3 / 4 6 2 → 0 · 4 / 1 9 10 3 → 2

Indicii (citește-le pe rând)

- «Prime între ele» înseamnă cmmdc egal cu 1. Deci ai nevoie de o funcție `Cmmdc(a, b)`.
- Algoritmul lui Euclid: cât timp `b != 0`, faci `r = a % b; a = b; b = r;`. La final, răspunsul e `a`.
- Parcurgi de la 0 până la `n - 2` (fără ultimul!) și numeri câte au `Cmmdc(v[i], ultim) == 1`.

Ideea, pe pași

- Funcția `Cmmdc` (Euclid).
- Citești vectorul și reții `ultim = v[n - 1]`.
- Pentru `i` de la 0 la `n - 2`: dacă `Cmmdc(v[i], ultim) == 1`, `cnt++`.
- Afișezi `cnt`.

BUCATA 1 DIN 3

```
using System;

class Program
{
    // cmmdc (algoritmul lui Euclid)
    static int Cmmdc(int a, int b)
    {
        while (b != 0)
        {
            int r = a % b;
            a = b;
            b = r;
        }
        return a;
    }
}
```

- 1 Euclid: împarți și păstrezi restul, iar și iar, până restul devine 0. Exemplu: `cmmdc(25, 12)`: $25 \% 12 = 1$, apoi $12 \% 1 = 0$, deci `cmmdc = 1`.

BUCATA 2 DIN 3

```
static void Main()
{
    char[] sep = { ' ', '\n', '\r', '\t' }; // separatori: spatiu, rand nou, tab
    string[] t = Console.In.ReadToEnd().Split(sep, StringSplitOptions.RemoveEmptyEntries);
    int k = 0;
    int n = int.Parse(t[k++]);
    int[] v = new int[n];
    for (int i = 0; i < n; i++)
        v[i] = int.Parse(t[k++]);
}
```

- 2 Citirea din șablon.

BUCATA 3 DIN 3

```
int ultim = v[n - 1];
int cnt = 0;
for (int i = 0; i < n - 1; i++) // toate, mai puțin ultimul
    if (Cmmdc(v[i], ultim) == 1) // "prime între ele" = cmmdc 1
        cnt++;

Console.WriteLine(cnt);
}
```

- 3 Ultimul element e pe poziția `n - 1` (pentru că pornim de la 0). Bucla merge cât timp `i < n - 1`, adică se oprește înainte de ultimul.

Ce a mers prost la încercarea ta: Eroare de compilare: `new int =[n]` (semnul `=` era în plus). Iar funcția ta de «prime între ele» nu era corectă; cu `cmmdc` e mult mai simplu.

Atenție la:

- «Prime între ele» $\neq$ «numere prime».
- Ultimul element e `v[n - 1]`, nu `v[n]`.
- Nu compara ultimul element cu el însuși.

Întrebări

15. Sunt 8 și 15 prime între ele?

- A. Da, $\text{cmmdc} = 1$
- B. Nu, pentru că 8 nu e prim
- C. Nu, $\text{cmmdc} = 3$
- D. Nu se poate ști

16. Cât e $\text{Cmmdc}(18, 12)$?

- A. 2
- B. 3
- C. 6
- D. 12

EgaleOglindit

Apare oglinditul primului număr?

Enunț rezumat. Textul complet: pbinfo.ro/probleme/294/egaleogclindit

Enunțul pe scurt

Primești n numere.

Calculezi **ogclinditul** primului număr, adică cifrele lui citite de la coadă (ogclinditul lui 123 e 321).

Verifici dacă ogclinditul apare **printre celelalte** numere (de la al doilea încolo). Afișezi **DA** sau **NU**.

Comparație. Scrii un număr pe o foaie și îl citești de la dreapta la stânga: 123 devine 321. Apoi te uiți dacă 321 e în restul listei.

PRIMEȘTI

n , apoi n numere naturale.

AFIȘEZI

DA dacă ogclinditul primului apare printre celelalte, altfel **NU**.

Exemplu

INTRARE

5
123 45 321 7 12

IEȘIRE

DA

De ce: Ogclinditul lui 123 e 321, iar 321 apare pe a treia poziție.

Exemplul de mai sus e făcut pentru ghid și verificat cu soluția; exemplul oficial e în enunțul de pe pbinfo.

Alte teste: 3 / 120 21 12 → DA · 3 / 56 56 6 → NU

Indicii (citește-le pe rând)

1. Ultima cifră a lui x e $x \% 10$. Ca s-o tai: $x /= 10$.
2. Construiești ogclinditul cu $r = r * 10 + x \% 10$; cât timp $x > 0$. Fiecare cifră nouă împinge cifrele de dinainte spre stânga.
3. Cauți ogclinditul de la $i = 1$ (primul element nu se ia în calcul). Folosește `long`, pentru că ogclinditul poate ieși mare.

Ideea, pe pași

1. Funcția `Ogclindit`.
2. Citești vectorul (cu `long`).
3. `og = Ogclindit(v[0])`.
4. Cauți `og` de la poziția 1 până la final.

BUCATA 1 DIN 3

```
using System;

class Program
{
    static long Oglindit(long x)
    {
        long r = 0;
        while (x > 0)
        {
            r = r * 10 + x % 10;
            x /= 10;
        }
        return r;
    }
}
```

- 1 Pentru 123: $r = 0 \cdot 10 + 3 = 3$, $x = 12$; $r = 3 \cdot 10 + 2 = 32$, $x = 1$; $r = 32 \cdot 10 + 1 = 321$, $x = 0$. Gata, $r = 321$.

BUCATA 2 DIN 3

```
static void Main()
{
    char[] sep = { ' ', '\n', '\r', '\t' }; // separatori: spatiu, rand nou, tab
    string[] t = Console.In.ReadToEnd().Split(sep, StringSplitOptions.RemoveEmptyEntries);
    int k = 0;
    int n = int.Parse(t[k++]);
    long[] v = new long[n];
    for (int i = 0; i < n; i++)
        v[i] = long.Parse(t[k++]);
}
```

- 2 La fel ca de obicei, doar că folosim `long` (numere întregi mai mari) și `long.Parse`. Oglinditul lui 1.000.000.009 ar fi 9.000.000.001, care nu încapă în `int`.

BUCATA 3 DIN 3

```
long og = Oglindit(v[0]);
bool gasit = false;
for (int i = 1; i < n; i++) // "printre CELELALTE" -> de la 1
    if (v[i] == og)
        gasit = true;

Console.WriteLine(gasit ? "DA" : "NU");
}
```

- 3 Căutare simplă cu un steguleț `gasit`. `gasit ? "DA" : "NU"` înseamnă: dacă `gasit` e true scrie DA, altfel NU.

La încercarea ta: 100 de puncte.

Atenție la:

- «Printre celelalte» = de la a doua poziție încolo.
- Oglinditul poate depăși `int`, deci folosește `long`.

Întrebări

17. Cât e oglinditul lui 4050?

- A. 0504
- B. 504
- C. 5040
- D. 4050

18. Care e formula de construire a oglinditului?

- A. $r = r + x \% 10$
- B. $r = r * 10 + x \% 10$
- C. $r = x / 10$
- D. $r = r * 10 + x / 10$

ElemPP

Numere care încep cu un pătrat perfect

Enunț rezumat. Textul complet: pbinfo.ro/probleme/132/elempp

Enunțul pe scurt

În `elempp.in` ai n numere naturale.

Scrii în `elempp.out`, **pe același rând**, numerele ale căror **primele două cifre** formează un pătrat perfect.

Pătratele perfecte de două cifre sunt: 16, 25, 36, 49, 64, 81 (adică $4 \cdot 4$, $5 \cdot 5$, ..., $9 \cdot 9$).

Comparație. Te uiți doar la «prefixul» numărului, ca la prefixul unui număr de telefon: din 1625 contează doar 16.

PRIMEȘTI

`elempp.in`: n , apoi n numere.

AFIȘEZI

`elempp.out`: numerele bune, pe un singur rând, separate prin spațiu.

Exemplu

ELEMPP.IN

6
1625 49 3612 777 2500 1234

ELEMPP.OUT

1625 49 3612 2500

De ce: Primele două cifre: 16 ✓, 49 ✓, 36 ✓, 77 ✗, 25 ✓, 12 ✗.

Exemplul de mai sus e făcut pentru ghid și verificat cu soluția; exemplul oficial e în enunțul de pe pbinfo.

Alte teste: `3 / 100 811 6400` → `811 6400`

Indicii (citește-le pe rând)

- Primele două cifre: împarți numărul la 10 cât timp e ≥ 100 . Din $3612 \rightarrow 361 \rightarrow 36$.
- Pătrat perfect: `int r = (int)Math.Sqrt(x);` și verifici dacă `r * r == x`.
- Afișează cu `fout.Write(x + " ")`, adică **Write**, nu **WriteLine**, ca să rămână totul pe același rând.

Ideea, pe pași

- Funcțiile `PrimeleDouaCifre` și `PatratPerfect`.
- Citești fiecare număr din fișier.
- Dacă primele lui două cifre formează pătrat perfect, îl scrii (cu `Write`).
- Închizi fișierul.

Soluția comentată (elempp.cs)

BUCATA 1 DIN 4

```
using System;
using System.IO;

class Program
{
    static bool PatratPerfect(int x)
    {
        int r = (int)Math.Sqrt(x);
        return r * r == x;
    }
}
```

- 1 `Math.Sqrt` dă radicalul (ex. $\sqrt{36} = 6$, $\sqrt{40} = 6,32\dots$). `(int)` taie zecimalele. Dacă $6 \cdot 6$ dă înapoi exact numărul, e pătrat perfect.

BUCATA 2 DIN 4

```
static int PrimeleDouaCifre(int x)
{
    while (x >= 100)        // taiem cifre de la final pana raman 2
        x /= 10;
    return x;
}
```

- 2 Cât timp numărul are 3 sau mai multe cifre (≥ 100), îi tăiem ultima cifră.

BUCATA 3 DIN 4

```
static void Main()
{
    char[] sep = { ' ', '\n', '\r', '\t' }; // separatori: spatiu, rand nou, tab
    string text = File.ReadAllText("elempp.in");
    string[] t = text.Split(sep, StringSplitOptions.RemoveEmptyEntries);
    int k = 0;
    int n = int.Parse(t[k++]);
}
```

- 3 Șablonul pentru fișiere.

BUCATA 4 DIN 4

```
StreamWriter fout = new StreamWriter("elempp.out");
for (int i = 0; i < n; i++)
{
    int x = int.Parse(t[k++]);
    if (PatratPerfect(PrimeleDouaCifre(x)))
        fout.Write(x + " "); // pe ACEEASI linie, separate prin spatiu
    fout.WriteLine();
    fout.Close();
}
}
```

- 4 Nici nu avem nevoie de vector: luăm fiecare număr și decidem pe loc. `PatratPerfect(PrimeleDouaCifre(x))` = întâi tăiem cifrele, apoi verificăm rezultatul.

Ce a mers prost la încercarea ta: Ai afișat fiecare număr pe rând nou (`fout.WriteLine`), dar rezultatul trebuia pe o singură linie. Write vs WriteLine e cea mai frecventă greșeală, iar aici a costat toate punctele.

Atenție la:

- «Pe același rând» → `Write`, nu `WriteLine`.
- `Math.Sqrt` întoarce un număr cu virgulă; îl transformi cu `(int)`.

Întrebări

19. Ce dă `PrimeleDouaCifre(98765)` ?

- A. 98
- B. 65
- C. 9876
- D. 87

20. Vrei 3 numere pe același rând în fișier. Ce folosești?

- A. `fout.WriteLine(x)`
- B. `fout.Write(x + " ")`
- C. `Console.Write(x)`
- D. `fout.Close(x)`

NrApPrime

Câte numere prime sunt în șir

Enunț rezumat. Textul complet: pbinfo.ro/probleme/100/nrappime

Enunțul pe scurt

În `nrappime.in` ai n numere.

Scrii în `nrappime.out` câte dintre ele sunt **prime**.

Un număr e prim dacă are exact doi divizori: 1 și el însuși (2, 3, 5, 7, 11, 13...). 0 și 1 NU sunt prime.

Comparație. Un număr prim e ca un grup de copii care nu se poate împărți în echipe egale (de cel puțin 2 copii fiecare): 7 copii îi poți pune doar 7 echipe de câte 1 sau o echipă de 7.

PRIMEȘTI

`nrappime.in`: n , apoi n numere naturale.

AFIȘEZI

`nrappime.out`: câte numere prime sunt.

Exemplu

NRAPPRIME.IN

6
2 9 1 13 25 17

NRAPPRIME.OUT

3

De ce: Prime: 2, 13, 17. Nu sunt prime: 9 ($=3 \cdot 3$), 1, 25 ($=5 \cdot 5$).

Exemplul de mai sus e făcut pentru ghid și verificat cu soluția; exemplul oficial e în enunțul de pe pbinfo.

Alte teste: 3 / 0 1 4 → 0 · 4 / 97 89 91 2 → 3

Indicii (citește-le pe rând)

1. Fă o funcție `EstePrim(x)` care întoarce `true` sau `false`. Primul lucru: dacă $x < 2$, nu e prim.
2. Încerci divizorii $d = 2, 3, 4, \dots$. Dacă $x \% d == 0$, ai găsit un divizor, deci nu e prim.
3. Nu trebuie să mergi până la x . E suficient cât timp $d * d \leq x$ (până la radical). Altfel programul e prea lent.

Ideea, pe pași

1. Funcția `EstePrim`.
2. Citești fiecare număr.
3. Dacă e prim, `cnt++`.
4. Scrii `cnt` în fișier.

BUCATA 1 DIN 3

```
using System;
using System.IO;

class Program
{
    static bool EstePrim(int x)
    {
        if (x < 2) return false;           // 0 și 1 nu sunt prime
        for (int d = 2; d * d <= x; d++)   // verificam doar până la radical
            if (x % d == 0)
                return false;
        return true;
    }
}
```

- 1 Dacă un număr are un divizor, are unul mai mic sau egal cu radicalul lui. De exemplu, la $91 = 7 \cdot 13$ îl găsim pe 7 ($7 \cdot 7 = 49 \leq 91$). De-asta ne oprim când $d * d > x$.

BUCATA 2 DIN 3

```
static void Main()
{
    char[] sep = { ' ', '\n', '\r', '\t' }; // separatori: spatiu, rand nou, tab
    string text = File.ReadAllText("nrappprime.in");
    string[] t = text.Split(sep, StringSplitOptions.RemoveEmptyEntries);
    int k = 0;
    int n = int.Parse(t[k++]);
}
```

- 2 Șablonul pentru fișiere.

BUCATA 3 DIN 3

```
int cnt = 0;
for (int i = 0; i < n; i++)
{
    int x = int.Parse(t[k++]);
    if (EstePrim(x))
        cnt++;
}

StreamWriter fout = new StreamWriter("nrappprime.out");
fout.WriteLine(cnt);
fout.Close();
}
```

- 3 Un contor: pornește de la 0 și crește la fiecare număr prim.

La încercarea ta: Ai ajuns la 100 după două încercări cu 0 puncte. Cele mai dese motive pentru 0 la numere prime: uiți că 0 și 1 nu sunt prime, sau cauți divizori până la x (prea lent).

Atenție la:

- 0 și 1 nu sunt prime.
- 2 este prim (singurul prim par).
- Divizori doar până la radical: $d * d \leq x$.

Întrebări

21. Pentru $x = 49$, care e ultimul d verificat de `for (int d = 2; d * d <= x; d++)` ?

- A. 6
- B. 7
- C. 48
- D. 49

22. Care dintre acestea e prim?

- A. 1
- B. 21
- C. 29
- D. 27

Prime4

Numărul prim care apare cel mai des

Enunț rezumat. Textul complet: pbinfo.ro/probleme/4456/prime4

Enunțul pe scurt

În `prime4.in` ai n numere, toate de **3 cifre** (între 100 și 999).

Găsești numărul **prim** care apare de **cele mai multe ori**. Dacă mai multe apar la fel de des, îl alegi pe **cel mai mic**.

Comparație. Ca la un vot: fiecare număr citit e un vot pentru el. Faci un tabel cu 1000 de căsuțe (câte una pentru fiecare număr posibil) și pui o liniuță în căsuța numărului de fiecare dată când apare. La final te uiți doar la căsuțele numerelor prime.

PRIMEȘTI

`prime4.in`: n , apoi n numere de 3 cifre.

AFIȘEZI

`prime4.out`: numărul prim cu cele mai multe apariții (la egalitate, cel mai mic).

Exemplu

PRIME4.IN

8
101 103 101 200 103 999 103 101

PRIME4.OUT

101

De ce: 101 apare de 3 ori, 103 tot de 3 ori. La egalitate îl alegem pe cel mai mic: 101. (200 și 999 nu sunt prime.)

Exemplul de mai sus e făcut pentru ghid și verificat cu soluția; exemplul oficial e în enunțul de pe pbinfo.

Alte teste: 5 / 997 997 113 113 113 → 113

Indicii (citește-le pe rând)

- Vector de frecvență: `int[] fr = new int[1000];` și pentru fiecare număr citit `fr[x]++`. Acum `fr[x]` spune de câte ori a apărut `x`.
- Parcurgi `x` de la 100 la 999, **crescător**, și te uiți doar la numerele prime care au apărut (`fr[x] > 0`).
- Schimbi răspunsul doar când găsești o frecvență **strict mai mare** (`>`). Așa, la egalitate rămâne cel mai mic, pentru că l-ai găsit primul.

Ideea, pe pași

- `fr[x]++` pentru fiecare număr citit.
- Parcurgi $x = 100 \dots 999$.
- Dacă x e prim, a apărut și are frecvență mai mare decât răspunsul curent, x devine răspuns.
- Scrii răspunsul.

Soluția comentată (prime4.cs)

BUCATA 1 DIN 4

```
using System;
using System.IO;

class Program
{
    static bool EstePrim(int x)
    {
        if (x < 2) return false;
        for (int d = 2; d * d <= x; d++)
            if (x % d == 0)
                return false;
        return true;
    }
}
```

- 1 Funcția clasică de număr prim (vezi NrApPrime).

BUCATA 2 DIN 4

```
static void Main()
{
    char[] sep = { ' ', '\n', '\r', '\t' }; // separatori: spatiu, rand nou, tab
    string text = File.ReadAllText("prime4.in");
    string[] t = text.Split(sep, StringSplitOptions.RemoveEmptyEntries);
    int k = 0;
    int n = int.Parse(t[k++]);
}
```

- 2 Șablonul pentru fișiere.

BUCATA 3 DIN 4

```
int[] fr = new int[1000]; // vector de FRECVENTA: fr[x] = de cate ori apare x
for (int i = 0; i < n; i++)
{
    int x = int.Parse(t[k++]);
    fr[x]++;
}
```

- 3 Trucul principal: numărul citit e folosit ca **poziție** în vector. Dacă citim 101, crește **fr[101]**. La început toate căsuțele sunt 0.

BUCATA 4 DIN 4

```
int rasp = -1;
// mergem crescator => la egalitate ramane cel mai mic
for (int x = 100; x <= 999; x++)
    if (EstePrim(x) && fr[x] > 0)
        if (rasp == -1 || fr[x] > fr[rasp])
            rasp = x;

StreamWriter fout = new StreamWriter("prime4.out");
fout.WriteLine(rasp);
fout.Close();
}
```

- 4 **rasp = -1** înseamnă «încă n-am găsit nimic». **&&** înseamnă «și». Folosim **>** strict: un număr mai mare, cu aceeași frecvență, NU îl înlocuiește pe cel găsit înainte.

La încercarea ta: 100 de puncte.

Atenție la:

- La egalitate vrei cel mai mic, deci parcurgi crescător și folosești `>`, nu `>=`.
- Vectorul de frecvență trebuie să aibă loc pentru 999, deci mărimea e 1000.

Întrebări

23. Se citesc 5, 3, 5, 5. Cât e `fr[5]` după citire?

- A. 1
- B. 2
- C. 3
- D. 5

24. Dacă ai folosi `>=` în loc de `>`, ce s-ar întâmpla la egalitate?

- A. Ar câștiga cel mai mic
- B. Ar câștiga cel mai mare, adică greșit
- C. Eroare de compilare
- D. Nimic diferit

inlocuireK

Cel mai apropiat multiplu de k, apoi afișare inversă

Enunț rezumat. Textul complet: pbinfo.ro/probleme/383/inlocuirek

Enunțul pe scurt

Primești **k** și **n** (pe același rând), apoi **n** numere.

Fiecare număr îl înlocuiești cu **cel mai apropiat multiplu al lui k**. Dacă e exact la mijloc între doi multipli, îl alegi pe **cel mai mic**.

Afișezi vectorul obținut **în ordine inversă** (de la ultimul la primul).

Comparație. E ca rotunjirea la cel mai apropiat 5 sau 10, doar că aici pasul e k. Cu k = 3, multiplii sunt 0, 3, 6, 9, 12... Numărul 7 e la 1 pas de 6 și la 2 pași de 9, deci devine 6.

PRIMEȘTI

Pe primul rând k și n, apoi n numere naturale.

AFIȘEZI

Vectorul transformat, de la coadă la început, pe un rând.

Exemplu

INTRARE

3 5
7 11 13 3 6

IEȘIRE

6 3 12 12 6

De ce: 7→6, 11→12, 13→12, 3→3, 6→6. Afișat invers: 6 3 12 12 6.

Exemplul de mai sus e făcut pentru ghid și verificat cu soluția; exemplul oficial e în enunțul de pe pbinfo.

Alte teste: 4 2 / 6 15 → 16 4

Indicii (citește-le pe rând)

- $r = x \% k$ îți spune cât de departe e x de multiplul de **dedesubt**, care este $x - r$.
- Multiplul de **deasupra** e $x - r + k$, iar distanța până la el e $k - r$.
- Dacă $r \leq k - r$, alegi multiplul de jos (la egalitate tot el câștigă), altfel pe cel de sus. Apoi afișezi cu **for** `(int i = n - 1; i >= 0; i--)`.

Ideea, pe pași

- Citești K și n din bucăți (sunt pe același rând!).
- Pentru fiecare element: calculezi **r**, **jos**, **sus** și alegi.
- Afișezi de la **n - 1** la 0.

Soluția comentată (inlocuirek.cs)

BUCATA 1 DIN 3

```
using System;

class Program
{
    static void Main()
    {
        char[] sep = { ' ', '\n', '\r', '\t' }; // separatori: spatiu, rand nou, tab
        string[] t = Console.In.ReadToEnd().Split(sep, StringSplitOptions.RemoveEmptyEntries);
        int k = 0;
        // k si n sunt pe ACEEASI linie -> de-asta citim pe bucati
        int K = int.Parse(t[k++]);
        int n = int.Parse(t[k++]);
        int[] v = new int[n];
        for (int i = 0; i < n; i++)
            v[i] = int.Parse(t[k++]);
    }
}
```

- 1 Atenție: `k` e deja «degetul» din șablon, așa că numărul `k` din enunț îl numim `K`. C# face diferența între litere mari și mici, deci sunt două variabile diferite.

BUCATA 2 DIN 3

```
for (int i = 0; i < n; i++)
{
    int r = v[i] % K;
    int jos = v[i] - r;           // multiplul de dedesubt
    int sus = jos + K;           // multiplul de deasupra
    if (r <= K - r)               // la egalitate il luam pe cel mai mic
        v[i] = jos;
    else
        v[i] = sus;
}
```

- 2 Exemplu cu $K = 3$ și $x = 11$: $r = 2$, $\text{jos} = 9$, $\text{sus} = 12$. Distanța până jos e 2, până sus e 1, deci alegem 12.

BUCATA 3 DIN 3

```
for (int i = n - 1; i >= 0; i--) // ordine inversa
    Console.Write(v[i] + " ");
Console.WriteLine();
}
```

- 3 Parcurgere de la coadă: pornim de la ultima poziție (`n - 1`) și scădem până la 0.

Ce a mers prost la încercarea ta: `k` și `n` sunt pe același rând, dar le-ai citit cu două `ReadLine` (programul crapă). Ai mai folosit `a[j]` fără să existe `j`, un `Console.Write()` gol și rotunjeai doar în jos.

Atenție la:

- Când două numere sunt pe același rând, `ReadLine` le ia pe amândouă ca un singur text, deci folosește șablonul.
- La egalitate alegi multiplul mai mic: `<=`, nu `<`.

Întrebări

25. $K = 5$, $x = 13$. Care e cel mai apropiat multiplu?

- A. 10
- B. 15
- C. 13
- D. 5

26. De ce nu merge `int K = int.Parse(Console.ReadLine());` când pe rând scrie "3 5"?

- A. Pentru că ReadLine citește tot rândul "3 5", care nu e un singur număr
- B. Pentru că 3 e prea mic
- C. Merge perfect
- D. Pentru că lipsește `using System.IO`

SumaCifreMax

Numărul cu cea mai mare sumă a cifrelor

Făcut pentru antrenament, în stilul problemelor de pe pbinfo. Soluția e verificată pe toate testele.

Enunțul pe scurt

Primești n numere naturale.

Afișezi numărul care are **cea mai mare sumă a cifrelor**. Dacă sunt mai multe, îl afișezi pe **primul**.

Comparație. E un maxim obișnuit, doar că nu compari numerele, ci un «scor» calculat pentru fiecare: suma cifrelor lui.

PRIMEȘTI

n , apoi n numere naturale.

AFIȘEZI

Numărul cu suma cifrelor maximă (primul, la egalitate).

Exemplu

INTRARE

5
123 99 405 18 900

IEȘIRE

99

De ce: Sumele: 123→6, 99→18, 405→9, 18→9, 900→9. Cea mai mare e 18, a lui 99.

Alte teste: 3 / 11 20 2 → 11 · 4 / 7 70 700 8 → 8

Indicii (citește-le pe rând)

1. Ai nevoie de funcția `SumaCifre(x)` (vezi Reguli → Funcții).
2. Ții minte **numărul** câștigător `rasp` și **suma** lui `best`. Pornești cu primul element.
3. Pentru fiecare număr următor: dacă suma lui e **strict mai mare** decât `best`, îl schimbi pe `rasp`.

Ideea, pe pași

1. Funcția `SumaCifre`.
2. `rasp = v[0]`, `best = SumaCifre(v[0])`.
3. Compari suma fiecărui element cu `best`.
4. Afișezi `rasp`.

Soluția comentată (sumacifremax.cs)

BUCATA 1 DIN 3

```
using System;

class Program
{
    static int SumaCifre(int x)
    {
        int s = 0;
        while (x > 0)
        {
            s += x % 10;
            x /= 10;
        }
        return s;
    }
}
```

- 1 Funcția clasică: ultima cifră cu `% 10`, o tăiem cu `/= 10`.

BUCATA 2 DIN 3

```
static void Main()
{
    char[] sep = { ' ', '\n', '\r', '\t' }; // separatori: spatiu, rand nou, tab
    string[] t = Console.In.ReadToEnd().Split(sep, StringSplitOptions.RemoveEmptyEntries);
    int k = 0;
    int n = int.Parse(t[k++]);
    int[] v = new int[n];
    for (int i = 0; i < n; i++)
        v[i] = int.Parse(t[k++]);
}
```

- 2 Citirea din șablon.

BUCATA 3 DIN 3

```
int rasp = v[0];
int best = SumaCifre(v[0]);
for (int i = 1; i < n; i++)
{
    int s = SumaCifre(v[i]);
    if (s > best) // > strict: la egalitate ramane primul
    {
        best = s;
        rasp = v[i];
    }
}

Console.WriteLine(rasp);
}
```

- 3 Un maxim cu două variabile: `best` e scorul, `rasp` e numărul care l-a obținut. Le schimbăm împreună.

Atenție la:

- Afișezi numărul, nu suma cifrelor lui.
- La egalitate rămâne primul, deci `>`, nu `>=`.

Întrebări

27. Cât e suma cifrelor lui 4096?

- A. 19
- B. 4096
- C. 13
- D. 10

28. Pentru `19 91 55`, ce afișează programul?

- A. 19
- B. 91
- C. 55
- D. 10

Palindroame

Câte numere sunt palindroame

Făcut pentru antrenament, în stilul problemelor de pe pbinfo. Soluția e verificată pe toate testele.

Enunțul pe scurt

În `palindroame.in` ai n numere naturale.

Scrii în `palindroame.out` câte dintre ele sunt **palindroame**, adică egale cu oglinditul lor (121, 7, 1221...).

Comparație. Un număr palindrom arată la fel și în oglindă: 1221 citit de la coadă e tot 1221.

PRIMEȘTI

`palindroame.in`: n , apoi n numere naturale.

AFIȘEZI

`palindroame.out`: câte numere sunt palindroame.

Exemplu

PALINDROAME.IN

6
121 34 7 1221 10 5665

PALINDROAME.OUT

4

De ce: 121, 7, 1221 și 5665 sunt palindroame. 34 (oglindit 43) și 10 (oglindit 1) nu sunt.

Alte teste: `3 / 10 20 30 → 0` · `2 / 0 11 → 2`

Indicii (citește-le pe rând)

1. Un număr e palindrom dacă e egal cu oglinditul lui. Ai nevoie de funcția `Oglindit`.
2. Oglinditul: `r = r * 10 + x % 10; x /= 10;` cât timp `x > 0`. Folosește `long`.
3. Pentru fiecare număr: dacă `Oglindit(x) == x`, `cnt++`. Scrii `cnt` în fișier.

Ideea, pe pași

1. Funcția `Oglindit`.
2. Citești din fișier.
3. Numeri câte sunt egale cu oglinditul.
4. Scrii numărul în fișier.

Soluția comentată (palindroame.cs)

BUCATA 1 DIN 3

```
using System;
using System.IO;

class Program
{
    static long Ogindit(long x)
    {
        long r = 0;
        while (x > 0)
        {
            r = r * 10 + x % 10;
            x /= 10;
        }
        return r;
    }
}
```

- 1 Aceeași funcție ca la EgaleOgindit.

BUCATA 2 DIN 3

```
static void Main()
{
    char[] sep = { ' ', '\n', '\r', '\t' }; // separatori: spatiu, rand nou, tab
    string text = File.ReadAllText("palindroame.in");
    string[] t = text.Split(sep, StringSplitOptions.RemoveEmptyEntries);
    int k = 0;
    int n = int.Parse(t[k++]);
}
```

- 2 Șablonul pentru fișiere.

BUCATA 3 DIN 3

```
int cnt = 0;
for (int i = 0; i < n; i++)
{
    long x = long.Parse(t[k++]);
    if (Ogindit(x) == x)
        cnt++;
}

StreamWriter fout = new StreamWriter("palindroame.out");
fout.WriteLine(cnt);
fout.Close();
}
```

- 3 Nu e nevoie de vector: verificăm fiecare număr imediat ce-l citim.

Atenție la:

- Oginditul lui 10 e 1, nu 01, deci 10 nu e palindrom.
- Nu uita `fout.Close()`.

Întrebări

29. Care dintre acestea e palindrom?

- A. 120
- B. 1001
- C. 1010
- D. 12

30. De ce folosim `long` pentru oglindit?

- A. Pentru că e mai rapid
- B. Pentru că oglinditul unui număr mare poate depăși `int`
- C. Pentru că `int` nu are cifre
- D. Nu e nevoie

6.3 Verificări (DA / NU)

Presupui că răspunsul e DA și cauți o dovadă că e NU.

De pe pbinfo: VerifDiferite (#502), PareOrdonate (#290), PalXXL (#2689), Identice (#505), Proportionale (#296).

Exerciții suplimentare: Crescator.

VerifDiferite

Sunt toate elementele diferite?

Enunț rezumat. Textul complet: pbinfo.ro/probleme/502/verifdiferite

Enunțul pe scurt

Primești n numere.

Afișezi **DA** dacă toate sunt **diferite două câte două** (nu există două egale), altfel **NU**.

Comparație. Ca la verificarea că într-o clasă nu există doi elevi cu același număr matricol: îl compari pe fiecare elev cu toți cei de după el.

PRIMEȘTI

n , apoi n numere.

AFIȘEZI

DA sau **NU**.

Exemplu

INTRARE

5

4 8 1 8 3

IEȘIRE

NU

De ce: 8 apare de două ori (pe pozițiile 2 și 4).

Exemplul de mai sus e făcut pentru ghid și verificat cu soluția; exemplul oficial e în enunțul de pe pbinfo.

Alte teste: 4 / 1 2 3 4 → **DA** · 1 / 7 → **DA**

Indicii (citește-le pe rând)

- Pornește de la presupunerea că răspunsul e DA: `bool diferite = true;`
- Compară fiecare pereche: i de la 0, j de la $i + 1$. Dacă `v[i] == v[j]`, poi `diferite = false`.
- Vectorul trebuie să fie `new int[n]`, nu o mărime fixă ca 200.

Ideea, pe pași

- Citești vectorul (mărime n !).
- `diferite = true`.
- Pentru fiecare pereche $i < j$: dacă sunt egale, `diferite = false`.
- Afișezi DA/NU.

BUCATA 1 DIN 2

```
using System;

class Program
{
    static void Main()
    {
        char[] sep = { ' ', '\n', '\r', '\t' }; // separatori: spatiu, rand nou, tab
        string[] t = Console.In.ReadToEnd().Split(sep, StringSplitOptions.RemoveEmptyEntries);
        int k = 0;
        int n = int.Parse(t[k++]);
        // marimea = n, NU un numar fix (200 era prea mic!)
        int[] v = new int[n];
        for (int i = 0; i < n; i++)
            v[i] = int.Parse(t[k++]);
    }
}
```

- 1 Vectorul are exact atâtea locuri câte numere sunt. Dacă scrii `new int[200]` și vin 500 de numere, programul crapă la al 201-lea.

BUCATA 2 DIN 2

```
bool diferite = true;
for (int i = 0; i < n; i++)
    for (int j = i + 1; j < n; j++)
        if (v[i] == v[j])
            diferite = false;

Console.WriteLine(diferite ? "DA" : "NU");
}
```

- 2 Tehnica «presupun că DA până găsesc o dovadă că NU». O singură pereche egală e suficientă ca să schimbe răspunsul.

Ce a mers prost la încercarea ta: Ai declarat `new int[200]`, dar `n` poate fi 500, așa că programul crăpa la testele mari (40 de puncte). Folosește mereu `new int[n]`.

Atenție la:

- Mărimea vectorului = `n`.
- Atenție la `==` (comparare) față de `=` (atribuire).

Întrebări

31. Ce se întâmplă cu `int[] v = new int[3];` dacă scrii în `v[3]`?

- A. Se adaugă automat un loc
- B. Programul crapă (IndexOutOfRangeException)
- C. Se scrie în `v[0]`
- D. Nimic

32. Care e diferența dintre `=` și `==` ?

- A. Nu e nicio diferență
- B. `=` pune o valoare, `==` compară
- C. `==` pune o valoare, `=` compară
- D. `==` e doar pentru text

PareOrdonate

Numerele pare sunt în ordine crescătoare?

Enunț rezumat. Textul complet: pbinfo.ro/probleme/290/pareordonate

Enunțul pe scurt

Primești n numere întregi (pot fi și negative).

Te uiți **doar la numerele pare**; pe cele impare le ignori. Afișezi **DA** dacă parele apar în ordine crescătoare (fiecare $\geq$ parul de dinainte), altfel **NU**.

Comparație. Urmărești doar jucătorii cu tricou roșu dintr-un șir: verifici că fiecare jucător roșu e cel puțin la fel de înalt ca jucătorul roșu dinaintea lui. Ceilalți jucători nu contează.

PRIMEȘTI

n , apoi n numere întregi.

AFIȘEZI

DA sau **NU**.

Exemplu

INTRARE

7
2 5 4 9 8 1 10

IEȘIRE

DA

De ce: Parele, în ordine: 2, 4, 8, 10. Cresc.

Exemplul de mai sus e făcut pentru ghid și verificat cu soluția; exemplul oficial e în enunțul de pe pbinfo.

Alte teste: 5 / 4 3 2 7 6 → **NU** · 4 / -4 -2 7 0 → **DA**

Indicii (citește-le pe rând)

- Ține minte **ultimul număr par** văzut (`ultimPar`) și dacă ai văzut deja unul (`amPar`).
- Când găsești un par nou: dacă `amPar` e true și `v[i] < ultimPar`, ordinea e stricată, deci `ok = false`.
- Testul de par `v[i] % 2 == 0` merge și pentru negative (`-4 % 2 == 0`). Dar atenție, `-3 % 2` dă `-1`, nu 1!

Ideea, pe pași

- Citești vectorul.
- Parcurgi; la fiecare par îl compari cu parul anterior.
- Dacă a scăzut, `ok = false`.
- Actualizezi `ultimPar`.

BUCATA 1 DIN 3

```
using System;

class Program
{
    static void Main()
    {
        char[] sep = { ' ', '\n', '\r', '\t' }; // separatori: spatiu, rand nou, tab
        string[] t = Console.In.ReadToEnd().Split(sep, StringSplitOptions.RemoveEmptyEntries);
        int k = 0;
        int n = int.Parse(t[k++]);
        int[] v = new int[n];
        for (int i = 0; i < n; i++)
            v[i] = int.Parse(t[k++]);
    }
}
```

1 Citirea din șablon.

BUCATA 2 DIN 3

```
bool ok = true;
bool amPar = false;
int ultimPar = 0; // ultimul numar par vazut
```

2 Trei variabile: răspunsul (**ok**), dacă am văzut deja un par (**amPar**) și care a fost ultimul par.

BUCATA 3 DIN 3

```
for (int i = 0; i < n; i++)
{
    if (v[i] % 2 == 0) // merge si pentru negative: -4 % 2 == 0
    {
        if (amPar && v[i] < ultimPar)
            ok = false; // a scazut => nu e crescator
        ultimPar = v[i];
        amPar = true;
    }
}

Console.WriteLine(ok ? "DA" : "NU");
}
```

3 Numerele impare nu intră în **if**, deci sunt ignorate. Primul par doar se ține minte (nu are cu cine fi comparat).

La încercarea ta: 100 de puncte.

Atenție la:

- Pentru impar la numere negative: **`x % 2 != 0`**, nu **`x % 2 == 1`**.
- Primul număr par n-are cu cine să fie comparat.

Întrebări

33. Cât dă $-7 \% 2$ în C#?

- A. 1
- B. -1
- C. 0
- D. 7

34. Șirul $6\ 1\ 6\ 3\ 8$: răspunsul?

- A. DA
- B. NU

PalXXL

Un număr uriaș: e palindrom?

Enunț rezumat. Textul complet: pbinfo.ro/probleme/2689/palxxl

Enunțul pe scurt

Primești n (până la 1000) și apoi **cifrele** unui număr, una câte una.

Pe primul rând afișezi numărul (cifrele lipite). Pe al doilea rând afișezi **DA** dacă numărul e **palindrom**, altfel **NU**.

Palindrom = se citește la fel de la stânga la dreapta și de la dreapta la stânga (ex: 12321).

Comparație. Ca la cuvintele «cojoc» sau «capac»: le citești de la ambele capete și ies la fel. Compari prima literă cu ultima, a doua cu penultima și tot așa.

PRIMEȘTI

n , apoi n cifre.

AFIȘEZI

Numărul, apoi pe rândul următor **DA** sau **NU**.

Exemplu

INTRARE

5

1 2 3 2 1

IEȘIRE

12321

DA

De ce: 1=1, 2=2, iar cifra din mijloc n -are pereche.

Exemplul de mai sus e făcut pentru ghid și verificat cu soluția; exemplul oficial e în enunțul de pe pbinfo.

Alte teste: 4 / 1 2 2 3 → 1223 / NU · 1 / 7 → 7 / DA

Indicii (citește-le pe rând)

1. Un număr cu 1000 de cifre nu încapă în `int` sau `long`. De aceea primești cifrele separat: le ții într-un vector.
2. Afișarea: `Console.Write(c[i])` pentru fiecare cifră (fără spații!), apoi `Console.WriteLine()`.
3. Palindrom: pentru `i` de la 0 cât timp `i < n / 2`, compari `c[i]` cu `c[n - 1 - i]`.

Ideea, pe pași

1. Citești cifrele în vector.
2. Le afișezi lipite.
3. Compari perechile din capete spre mijloc.
4. Afișezi DA/NU.

Soluția comentată (palxxl.cs)

BUCATA 1 DIN 3

```
using System;

class Program
{
    static void Main()
    {
        char[] sep = { ' ', '\n', '\r', '\t' }; // separatori: spatiu, rand nou, tab
        string[] t = Console.In.ReadToEnd().Split(sep, StringSplitOptions.RemoveEmptyEntries);
        int k = 0;
        int n = int.Parse(t[k++]);
        int[] c = new int[n];
        for (int i = 0; i < n; i++)
            c[i] = int.Parse(t[k++]);
    }
}
```

- 1 Citim cifrele într-un vector `c`.

BUCATA 2 DIN 3

```
// numarul poate avea 1000 de cifre -> NU il
// putem pune intr-un int, il afisam cifra cu cifra
for (int i = 0; i < n; i++)
    Console.Write(c[i]);
Console.WriteLine();
```

- 2 `Write` fără spațiu → cifrele apar lipite: 12321.

BUCATA 3 DIN 3

```
bool pal = true;
for (int i = 0; i < n / 2; i++)
    if (c[i] != c[n - 1 - i]) // prima cu ultima, a doua cu penultima...
        pal = false;

Console.WriteLine(pal ? "DA" : "NU");
}
```

- 3 Perechea lui `i` e `n - 1 - i`: pentru `i = 0` e ultima poziție, pentru `i = 1` e penultima. Mergem doar până la jumătate (`n / 2`).

La încercarea ta: 100 de puncte.

Atenție la:

- Cifrele se afișează lipite (fără spațiu).
- Perechea lui `i` este `n - 1 - i`.

Întrebări

35. Pentru $n = 6$, cu ce poziție se compară poziția 1?

- A. 5
- B. 4
- C. 6
- D. 1

36. Câte **7 / 2** în C# (cu int)?

- A. 3.5
- B. 3
- C. 4
- D. eroare

Identice

Doi vectori cu aceleași elemente?

Enunț rezumat. Textul complet: pbinfo.ro/probleme/505/identice

Enunțul pe scurt

Primești n , apoi doi vectori a și b , fiecare cu n numere.

Afișezi **DA** dacă cei doi conțin **exact aceleași elemente** (de același număr de ori), chiar dacă în altă ordine. Altfel **NU**.

Comparație. Ai două pachete de cărți amestecate. Ca să vezi dacă au aceleași cărți, le sortezi pe amândouă și apoi compari carte cu carte.

PRIMEȘTI

n , apoi n numere (vectorul a), apoi n numere (vectorul b).

AFIȘEZI

DA sau **NU**.

Exemplu

INTRARE

4
3 1 2 3
2 3 3 1

IEȘIRE

DA

De ce: Sortați: $a = 1\ 2\ 3\ 3$ și $b = 1\ 2\ 3\ 3$. Sunt identici.

Exemplul de mai sus e făcut pentru ghid și verificat cu soluția; exemplul oficial e în enunțul de pe pbinfo.

Alte teste: 3 / 1 2 2 / 1 1 2 → **NU**

Indicii (citește-le pe rând)

1. Truc: dacă doi vectori au aceleași elemente, după sortare devin **identici**, poziție cu poziție.
2. Sortarea în C#: `Array.Sort(a);`, o singură linie!
3. Compari `a[i]` cu `b[i]` pentru fiecare `i`. La prima diferență răspunsul e **NU**.

Ideea, pe pași

1. Citești a și b .
2. Sortezi ambele.
3. Compari poziție cu poziție.
4. Afișezi DA/NU.

Soluția comentată (identice.cs)

BUCATA 1 DIN 3

```
using System;

class Program
{
    static void Main()
    {
        char[] sep = { ' ', '\n', '\r', '\t' }; // separatori: spatiu, rand nou, tab
        string[] t = Console.In.ReadToEnd().Split(sep, StringSplitOptions.RemoveEmptyEntries);
        int k = 0;
        int n = int.Parse(t[k++]);
        int[] a = new int[n];
        int[] b = new int[n];
        for (int i = 0; i < n; i++) a[i] = int.Parse(t[k++]);
        for (int i = 0; i < n; i++) b[i] = int.Parse(t[k++]);
    }
}
```

- 1 Doi vectori citiți unul după altul. Degetul **k** continuă de unde a rămas.

BUCATA 2 DIN 3

```
// dupa sortare, vectori cu aceleasi elemente devin IDENTICI
Array.Sort(a);
Array.Sort(b);
```

- 2 **Array.Sort** ordonează crescător, direct în vector.

BUCATA 3 DIN 3

```
bool ok = true;
for (int i = 0; i < n; i++)
    if (a[i] != b[i])
    {
        ok = false;
        break;
    }

Console.WriteLine(ok ? "DA" : "NU");
}
```

- 3 **!=** înseamnă «diferit». La prima diferență ne oprim cu **break**. Observă că **break** e ÎN acolade, împreună cu **ok = false**.

Ce a mers prost la încercarea ta: Eroare de compilare: lipsea o paranteză la **int.Parse(s[i]**, în for ai scris **k++** în loc de **i++**, iar **break** era în afara acoladelor.

Atenție la:

- Fiecare **(** are nevoie de **)**, fiecare **{** de **}**.
- În for, crește variabila buclei (**i++**), nu alta.

Întrebări

37. Ce face `Array.Sort(v)` pe `5 1 4` ?

- A. 5 4 1
- B. 1 4 5
- C. 1 5 4
- D. Afișează 1 4 5

38. Unde trebuie pus `break` ca să se execute doar când găsim o diferență?

- A. După acolada if-ului
- B. În acoladele if-ului, după `ok = false`
- C. Înainte de for
- D. Nu contează

Proportionale

Pot deveni șirurile direct proporționale?

Enunț rezumat. Textul complet: pbinfo.ro/probleme/296/proportionale

Enunțul pe scurt

Primești n , apoi două șiruri a și b , cu câte n numere naturale nenule.

Întrebarea: poți rearanja numerele din fiecare șir astfel încât să fie **direct proporționale**? Adică $a[0]/b[0] = a[1]/b[1] = \dots$, același raport peste tot. Afișezi **DA** sau **NU**.

Comparație. Ca la o rețetă: la 2 ouă pui 3 linguri de zahăr, la 4 ouă pui 6, la 6 ouă pui 9. Cel mai mic merge cu cel mai mic, cel mai mare cu cel mai mare.

PRIMEȘTI

n , apoi șirul a , apoi șirul b .

AFIȘEZI

DA sau **NU**.

Exemplu

INTRARE

3
2 6 4
9 3 6

IEȘIRE

DA

De ce: Sortate: $a = 2\ 4\ 6$ și $b = 3\ 6\ 9$. Rapoartele $2/3$, $4/6$, $6/9$ sunt toate egale cu $2/3$.

Exemplul de mai sus e făcut pentru ghid și verificat cu soluția; exemplul oficial e în enunțul de pe pbinfo.

Alte teste: 3 / 1 2 3 / 2 4 7 → **NU**

Indicii (citește-le pe rând)

1. Dacă se poate, atunci cel mai mic din a merge cu cel mai mic din b și tot așa. Deci **sortezi** ambele.
2. Nu împărți! $2 / 3$ cu `int` dă 0. Folosește «produsul pe cruce»: $a[i]/b[i] == a[0]/b[0]$ e același lucru cu $a[i]*b[0] == b[i]*a[0]$.
3. Produsul poate ieși mare, așa că îl calculezi în `long`: $(long)a[i] * b[0]$.

Ideea, pe pași

1. Citești a și b .
2. Sortezi ambele.
3. Pentru fiecare i , verifici produsul pe cruce cu perechea 0.
4. Afișezi DA/NU.

Soluția comentată (proportionale.cs)

BUCATA 1 DIN 3

```
using System;

class Program
{
    static void Main()
    {
        char[] sep = { ' ', '\n', '\r', '\t' }; // separatori: spatiu, rand nou, tab
        string[] t = Console.In.ReadToEnd().Split(sep, StringSplitOptions.RemoveEmptyEntries);
        int k = 0;
        int n = int.Parse(t[k++]);
        int[] a = new int[n];
        int[] b = new int[n];
        for (int i = 0; i < n; i++) a[i] = int.Parse(t[k++]);
        for (int i = 0; i < n; i++) b[i] = int.Parse(t[k++]);
    }
}
```

- 1 Citim cei doi vectori.

BUCATA 2 DIN 3

```
// sortate, cel mai mic merge cu cel mai mic, al 2-lea cu al 2-lea etc.
Array.Sort(a);
Array.Sort(b);
```

- 2 După sortare, perechile «potrivite» ajung pe aceleași poziții.

BUCATA 3 DIN 3

```
// a[i]/b[i] == a[0]/b[0] <=> a[i]*b[0] == b[i]*a[0] (fara impartiri cu virgula!)
bool ok = true;
for (int i = 1; i < n; i++)
    if ((long)a[i] * b[0] != (long)b[i] * a[0])
        ok = false;

Console.WriteLine(ok ? "DA" : "NU");
}
```

- 3 Ca la fracții la mate: $2/3 = 4/6$ pentru că $2 \cdot 6 = 3 \cdot 4$. `(long)` transformă numărul în `long` înainte de înmulțire, ca să nu depășească limita lui `int` (~2,1 miliarde).

La încercarea ta: 100 de puncte.

Atenție la:

- Împărțirea între int-uri taie zecimalele: `2 / 3 == 0`.
- Înmulțirea a două numere mari poate depăși `int`, deci folosește `(long)`.

Întrebări

39. Sunt $3/4$ și $9/12$ egale? Cum verifici fără împărțire?

- A. $3 \cdot 12 == 4 \cdot 9 \rightarrow 36 == 36 \rightarrow$ DA
- B. $3 + 12 == 4 + 9 \rightarrow$ NU
- C. $3 \cdot 4 == 9 \cdot 12 \rightarrow$ NU
- D. Nu se poate fără împărțire

40. Cât dă $(5 / 2) * 2$ cu int?

- A. 5
- B. 4
- C. 5.0
- D. 6

Crescator

Este vectorul strict crescător?

Făcut pentru antrenament, în stilul problemelor de pe pbinfo. Soluția e verificată pe toate testele.

Enunțul pe scurt

Primești n numere întregi.

Afișezi **DA** dacă vectorul e **strict crescător** (fiecare element e mai mare decât cel dinainte), altfel **NU**.

Comparație. Ca o scară pe care urci mereu: fiecare treaptă trebuie să fie mai sus decât precedentă. O treaptă la fel de înaltă sau mai joasă strică totul.

PRIMEȘTI

n , apoi n numere întregi.

AFIȘEZI

DA sau **NU**.

Exemplu

INTRARE

5
2 4 7 9 15

IEȘIRE

DA

Alte teste: 4 / 1 3 3 5 → **NU** · 1 / 42 → **DA**

Indicii (citește-le pe rând)

1. Compari fiecare element cu vecinul din stânga: $v[i]$ cu $v[i - 1]$.
2. Pornești cu `ok = true`. Dacă găsești $v[i] \leq v[i - 1]$, pui `ok = false`.
3. For-ul pornește de la `i = 1`, ca $v[i - 1]$ să existe.

Ideea, pe pași

1. Citești vectorul.
2. `ok = true`.
3. Pentru i de la 1: dacă $v[i] \leq v[i-1]$, `ok = false`.
4. Afișezi DA/NU.

BUCATA 1 DIN 2

```
using System;

class Program
{
    static void Main()
    {
        char[] sep = { ' ', '\n', '\r', '\t' }; // separatori: spatiu, rand nou, tab
        string[] t = Console.In.ReadToEnd().Split(sep, StringSplitOptions.RemoveEmptyEntries);
        int k = 0;
        int n = int.Parse(t[k++]);
        int[] v = new int[n];
        for (int i = 0; i < n; i++)
            v[i] = int.Parse(t[k++]);
    }
}
```

1 Citirea din șablon.

BUCATA 2 DIN 2

```
bool ok = true;
for (int i = 1; i < n; i++)
    if (v[i] <= v[i - 1]) // egal sau mai mic -> nu e strict crescator
        ok = false;

Console.WriteLine(ok ? "DA" : "NU");
}
```

2 Fiecare pereche de vecini trebuie să «urce». Dacă măcar o pereche nu urcă, răspunsul e NU.

Atenție la:

- «Strict» crescător: egalitatea strică ordinea, deci `<=`.
- Pornești de la `i = 1`.

Întrebări

41. Este `3 5 5 8` strict crescător?

- A. DA
- B. NU

42. Ce se întâmplă dacă for-ul pornește de la `i = 0`?

- A. Nimic
- B. Se accesează `v[-1]` și programul crapă
- C. Răspunsul e mereu DA
- D. Merge mai repede

6.4 Sortare

`Array.Sort` ordonează crescător; de la coadă obții ordinea descrescătoare.

De pe pbinfo: Sort10 (#4146). Exerciții suplimentare: KMici.

Sort10

Multiplii de 10, în ordine descrescătoare

Enunț rezumat. Textul complet: pbinfo.ro/probleme/4146/sort10

Enunțul pe scurt

Primești n numere.

Afișezi, **pe același rând**, în ordine **descrescătoare**, numerele divizibile cu 10. Dacă nu există niciunul, afișezi **NU EXISTA**.

Comparație. Scoți din portofel doar bancnotele «rotunde» (10, 50, 100...) și le așezi de la cea mai mare la cea mai mică.

PRIMEȘTI

n , apoi n numere naturale.

AFIȘEZI

Multiplii de 10 descrescător, pe un rând, sau **NU EXISTA**.

Exemplu

INTRARE

6
40 7 100 15 20 30

IEȘIRE

100 40 30 20

De ce: Divizibile cu 10: 40, 100, 20, 30. Descrescător: 100 40 30 20.

Exemplul de mai sus e făcut pentru ghid și verificat cu soluția; exemplul oficial e în enunțul de pe pbinfo.

Alte teste: 3 / 1 2 3 → **NU EXISTA** · 4 / 10 10 5 70 → **70 10 10**

Indicii (citește-le pe rând)

- `Array.Sort(v)` sortează **crescător**. Pentru descrescător, îl parcurgi de la coadă: `for (int i = n - 1; i >= 0; i--)`.
- Divizibil cu 10 înseamnă `v[i] % 10 == 0`.
- Toate pe același rând: `Console.Write(v[i] + " ")`. Ține un `bool exista` ca să știi dacă trebuie să scrii **NU EXISTA**.

Ideea, pe pași

- Citești.
- Sortezi crescător.
- Parcurgi de la coadă și afișezi multiplii de 10.
- Dacă n-ai afișat nimic: **NU EXISTA**.

Soluția comentată (sort10.cs)

BUCATA 1 DIN 4

```
using System;

class Program
{
    static void Main()
    {
        char[] sep = { ' ', '\n', '\r', '\t' }; // separatori: spatiu, rand nou, tab
        string[] t = Console.In.ReadToEnd().Split(sep, StringSplitOptions.RemoveEmptyEntries);
        int k = 0;
        int n = int.Parse(t[k++]);
        int[] v = new int[n];
        for (int i = 0; i < n; i++)
            v[i] = int.Parse(t[k++]);
    }
}
```

1 Citirea din șablon.

BUCATA 2 DIN 4

```
Array.Sort(v); // crescator...
```

2 După această linie, vectorul e ordonat crescător.

BUCATA 3 DIN 4

```
bool exista = false;
for (int i = n - 1; i >= 0; i--) // ...si il parcurgem de la coada => descrescator
    if (v[i] % 10 == 0)
    {
        Console.Write(v[i] + " "); // Write, NU WriteLine (toate pe acelasi rand)
        exista = true;
    }
```

3 Citit de la coadă, un vector crescător devine descrescător. Afișăm doar ce se împarte exact la 10 (restul e 0).

BUCATA 4 DIN 4

```
if (!exista)
    Console.WriteLine("NU EXISTA");
Console.WriteLine();
}
```

4 Dacă steagul a rămas `false`, n-am afișat nimic, deci scriem mesajul din enunț.

Ce a mers prost la încercarea ta: Ai folosit `Console.WriteLine(x + " ")`, deci fiecare număr apărea pe alt rând. Trebuia `Console.Write`. Logica era bună: ai pierdut 80 de puncte doar din asta.

Atenție la:

- «Pe același rând» → `Write`.
- Descrescător = sortare crescătoare + parcurgere de la coadă.

Întrebări

43. $v = 10\ 30\ 20$ după `Array.Sort(v)`. Ce afișează parcurgerea de la `n - 1` la 0?

- A. 10 20 30
- B. 30 20 10
- C. 10 30 20
- D. 20 30 10

44. Ce afișează `Console.WriteLine(1); Console.WriteLine(2);`?

- A. 1 2
- B. 12
- C. 1 și 2 pe rânduri separate
- D. Eroare

KMici

Cele mai mici k numere, crescător

Făcut pentru antrenament, în stilul problemelor de pe pbinfo. Soluția e verificată pe toate testele.

Enunțul pe scurt

Primești **n** și **k** (pe același rând), apoi **n** numere.

Afișezi, pe un rând, **cele mai mici k** numere, în ordine **crescătoare** (dacă o valoare apare de mai multe ori, se ia de fiecare dată).

Comparație. Ca la o probă de alergare: aranjezi timpii de la cel mai bun la cel mai slab și citești doar primii k.

PRIMEȘTI

n și k, apoi n numere ($k \leq n$).

AFIȘEZI

Primele k valori din vectorul sortat crescător.

Exemplu

INTRARE

6 3
40 7 100 15 20 7

IEȘIRE

7 7 15

De ce: Sortat: 7 7 15 20 40 100. Primele 3: 7 7 15.

Alte teste: 3 3 / 5 1 4 → 1 4 5

Indicii (citește-le pe rând)

1. Sortezi vectorul crescător cu `Array.Sort(v)`.
2. Atunci cele mai mici k numere sunt chiar primele k: `v[0] ... v[k - 1]`.
3. Afișezi cu `for (int i = 0; i < k; i++) Console.Write(v[i] + " ");`.

Ideea, pe pași

1. Citești n, k și vectorul.
2. `Array.Sort(v)`.
3. Afișezi primele k elemente pe un rând.

Soluția comentată (kmici.cs)

BUCATA 1 DIN 2

```
using System;

class Program
{
    static void Main()
    {
        char[] sep = { ' ', '\n', '\r', '\t' }; // separatori: spatiu, rand nou, tab
        string[] t = Console.In.ReadToEnd().Split(sep, StringSplitOptions.RemoveEmptyEntries);
        int p = 0; // "degetul" (k e deja numele din enunt)
        int n = int.Parse(t[p++]);
        int k = int.Parse(t[p++]);
        int[] v = new int[n];
        for (int i = 0; i < n; i++)
            v[i] = int.Parse(t[p++]);
    }
}
```

- 1 Aici enunțul folosește deja numele **k**, așa că degetul din șablon îl numim **p**. Orice nume merge, important e să nu se repete.

BUCATA 2 DIN 2

```
Array.Sort(v); // crescator

for (int i = 0; i < k; i++)
    Console.Write(v[i] + " ");
Console.WriteLine();
}
```

- 2 După sortare, cele mai mici valori sunt la început. Le afișăm doar pe primele k, pe același rând.

Atenție la:

- Bucula de afișare merge până la **k**, nu până la **n**.
- Nu folosi același nume pentru deget și pentru k.

Întrebări

45. $v = 9\ 2\ 6\ 2$, $k = 2$. Ce se afișează?

- A. 2 6
- B. 2 2
- C. 9 6
- D. 2 9

46. De ce degetul se numește **p** și nu **k** aici?

- A. Așa cere C#
- B. Pentru că **k** e deja numele unei date din enunț și n-ai voie două variabile cu același nume
- C. Pentru că p e mai scurt
- D. Din greșeală

6.5 Secvențe

Bucăți de elemente vecine: le numeri sau le cauți după sumă.

De pe pbinfo: Secvente (#304), SumaInSecv (#300). Exerciții suplimentare: SecvMax.

Secvente

Câte bucăți crescătoare are șirul

Enunț rezumat. Textul complet: pbinfo.ro/probleme/304/secvente

Enunțul pe scurt

În `secvente.in` ai n numere (pot fi scrise pe mai multe rânduri).

O **secvență** e o bucată de elemente **vecine**. E crescătoare dacă fiecare element e $\geq$ cel dinainte.

Tai șirul în bucăți crescătoare cât mai lungi (maximale) și scrii în `secvente.out` **câte** bucăți ies.

Comparație. Ca la urcatul pe scări: cât timp urci (sau stai pe loc), ești în aceeași «urcare». Când cobori o treaptă, începe o urcare nouă. Numeri câte urcări ai făcut.

PRIMEȘTI

`secvente.in`: n , apoi n numere.

AFIȘEZI

`secvente.out`: numărul de secvențe crescătoare maximale.

Exemplu

SECVENTE.IN

8
1 3 3 2 5 7 4 4

SECVENTE.OUT

3

De ce: Bucățile: [1 3 3] [2 5 7] [4 4]. O bucată nouă începe la 2 (pentru că $2 < 3$) și la 4 (pentru că $4 < 7$).

Exemplul de mai sus e făcut pentru ghid și verificat cu soluția; exemplul oficial e în enunțul de pe pbinfo.

Alte teste: 4 / 5 4 3 2 → 4 · 3 / 1 2 3 → 1

Indicii (citește-le pe rând)

1. Nu trebuie să găsești efectiv bucățile. Întreabă-te doar: când începe o bucată nouă?
2. O bucată nouă începe exact când șirul **scade**: `v[i] < v[i - 1]`.
3. Pornești cu `cnt = 1` (prima bucată) și aduni 1 la fiecare scădere.

Ideea, pe pași

1. Citești.
2. `cnt = 1`.
3. Pentru i de la 1: dacă `v[i] < v[i-1]`, `cnt++`.
4. Scrii `cnt`.

Soluția comentată (secvente.cs)

BUCATA 1 DIN 3

```
using System;
using System.IO;

class Program
{
    static void Main()
    {
        char[] sep = { ' ', '\n', '\r', '\t' }; // separatori: spatiu, rand nou, tab
        string text = File.ReadAllText("secvente.in");
        string[] t = text.Split(sep, StringSplitOptions.RemoveEmptyEntries);
        int k = 0;
        int n = int.Parse(t[k++]);
        int[] v = new int[n];
        for (int i = 0; i < n; i++)
            v[i] = int.Parse(t[k++]);
    }
}
```

- 1 Citirea din fișier. Nu contează pe câte rânduri sunt numerele: `Split` le separă oricum.

BUCATA 2 DIN 3

```
// o secventa noua incepe de fiecare data cand sirul SCADDE
int cnt = 1;
for (int i = 1; i < n; i++)
    if (v[i] < v[i - 1])
        cnt++;
```

- 2 Pornim de la `i = 1` pentru că îl comparăm pe fiecare cu vecinul din stânga (`v[i - 1]`). Pentru `i = 0` n-ar exista vecin.

BUCATA 3 DIN 3

```
StreamWriter fout = new StreamWriter("secvente.out");
fout.WriteLine(cnt);
fout.Close();
}
```

- 3 Scriem rezultatul și închidem fișierul.

La încercarea ta: 100 de puncte. Atenție: numerele pot fi pe mai multe rânduri, deci citirea pe bucăți (șablonul) e obligatorie.

Atenție la:

- Comparăția cu vecinul pornește de la `i = 1`.
- Egalitatea NU începe o bucată nouă.

Întrebări

47. Câte secvențe crescătoare maxime are `5 6 2 2 9 1`?

- A. 2
- B. 3
- C. 4
- D. 6

48. De ce for-ul pornește de la `i = 1`?

- A. Ca să sară peste primul număr
- B. Pentru că îl comparăm cu $v[i - 1]$, iar $v[-1]$ nu există
- C. Din greșeală
- D. Pentru că indicii încep de la 1

SumainSecv

Prima bucată de vecini cu suma S

Enunț rezumat. Textul complet: pbinfo.ro/probleme/300/sumainsecv

Enunțul pe scurt

În `sumainsecv.in` ai n și S , apoi n numere naturale nenule.

Cauți o **secvență** (elemente vecine) cu suma exact S . Dacă sunt mai multe, o vrei pe cea care începe **cel mai în stânga**.

Scrii pozițiile de început și de sfârșit (numerotate de la 1). Dacă nu există, scrii `0 0`.

Comparație. Ai un șir de monede pe masă și vrei să iei monede **vecine** care să facă fix S lei. Pornești de la prima monedă și tot adaugi monede din dreapta. Dacă ai trecut de S , încerci de la următoarea monedă.

PRIMEȘTI

`sumainsecv.in`: n și S , apoi n numere.

AFIȘEZI

`sumainsecv.out`: pozițiile p și u (de la 1) sau `0 0`.

Exemplu

SUMAINSECV.IN

7 10

2 5 1 4 6 3 7

SUMAINSECV.OUT

2 4

De ce: De la poziția 1: 2, 7, 8, 12, am trecut de 10. De la poziția 2: 5, 6, 10, gata! Pozițiile 2...4 ($5 + 1 + 4 = 10$).

Exemplul de mai sus e făcut pentru ghid și verificat cu soluția; exemplul oficial e în enunțul de pe pbinfo.

Alte teste: `3 100 / 1 2 3 → 0 0` · `4 6 / 6 1 2 3 → 1 1`

Indicii (citește-le pe rând)

- Încerci fiecare început posibil i (de la 1 la n), de la stânga la dreapta.
- Pentru un început i , aduni pe rând $v[i]$, $v[i+1]$, ... și verifici suma după fiecare adunare.
- Dacă suma == S , ai găsit. Dacă suma > S , te oprești, pentru că numerele sunt pozitive și suma doar crește.

Ideea, pe pași

- Citești n , S și vectorul în `v[1..n]`.
- Pentru fiecare început i : suma = 0, aduni spre dreapta.
- La suma == S , salvezi p , u și te oprești de tot.
- Scrii p și u (rămân 0 0 dacă n-ai găsit).

Soluția comentată (sumainsecv.cs)

BUCATA 1 DIN 3

```
using System;
using System.IO;

class Program
{
    static void Main()
    {
        char[] sep = { ' ', '\n', '\r', '\t' }; // separatori: spatiu, rand nou, tab
        string text = File.ReadAllText("sumainsecv.in");
        string[] t = text.Split(sep, StringSplitOptions.RemoveEmptyEntries);
        int k = 0;
        int n = int.Parse(t[k++]);
        int S = int.Parse(t[k++]);
        int[] v = new int[n + 1]; // indici de la 1
        for (int i = 1; i <= n; i++)
            v[i] = int.Parse(t[k++]);
    }
}
```

1 Indici de la 1, pentru că răspunsul se cere cu poziții de la 1.

BUCATA 2 DIN 3

```
int p = 0, u = 0;
// incepem de la stanga => prima gasita e cea mai din stanga
for (int i = 1; i <= n && p == 0; i++)
{
    int suma = 0;
    for (int j = i; j <= n; j++)
    {
        suma += v[j];
        if (suma == S) { p = i; u = j; break; }
        // numerele sunt pozitive, suma doar creste
        if (suma > S) break;
    }
}
```

2 `p == 0` în condiția for-ului mare înseamnă «continuă doar cât timp n-am găsit». `break` din for-ul mic oprește doar for-ul mic; for-ul mare se oprește singur pentru că `p` nu mai e 0.

BUCATA 3 DIN 3

```
StreamWriter fout = new StreamWriter("sumainsecv.out");
fout.WriteLine(p + " " + u); // daca nu s-a gasit, raman 0 0
fout.Close();
}
```

3 Dacă n-am găsit nimic, p și u au rămas 0, deci se scrie automat `0 0`.

La încercarea ta: 100 de puncte.

Atenție la:

- Pozițiile se afișează de la 1.
- `break` iese doar din bucla în care se află.
- Valorile inițiale `p = u = 0` rezolvă singure cazul «nu există».

Întrebări

49. $n = 4$, $S = 5$, $v = 1\ 1\ 3\ 5$. Ce se afișează?

- A. 1 3
- B. 2 3
- C. 4 4
- D. 0 0

50. Avem două for-uri unul în altul. `break` din cel interior...

- A. oprește ambele
- B. oprește doar for-ul interior
- C. oprește programul
- D. nu face nimic

SecvMax

Cea mai lungă secvență de elemente egale

Făcut pentru antrenament, în stilul problemelor de pe pbinfo. Soluția e verificată pe toate testele.

Enunțul pe scurt

În `secvmax.in` ai `n` numere.

Scrii în `secvmax.out` **lungimea** celei mai lungi secvențe formate din elemente **vecine egale**.

Comparație. Ca la un joc în care numeri «seria» de aruncări la fel: când apare altceva, seria se rupe și o iei de la 1. Ții minte cea mai lungă serie.

PRIMEȘTI

`secvmax.in`: `n`, apoi `n` numere.

AFIȘEZI

`secvmax.out`: lungimea maximă.

Exemplu

SECVMAX.IN

9
3 3 5 5 5 2 5 5 1

SECVMAX.OUT

3

De ce: Seriile: [3 3] [5 5 5] [2] [5 5] [1]. Cea mai lungă are 3 elemente.

Alte teste: 4 / 1 2 3 4 → 1 · 5 / 7 7 7 7 7 → 5

Indicii (citește-le pe rând)

- Ții două numere: `lung` (lungimea seriei curente) și `maxim` (cea mai lungă de până acum). Amândouă pornesc de la 1.
- Pentru fiecare `i` de la 1: dacă `v[i] == v[i - 1]`, seria continuă (`lung++`); altfel începe una nouă (`lung = 1`).
- După fiecare pas: dacă `lung > maxim`, `maxim = lung`.

Ideea, pe pași

- Citești vectorul.
- `lung = maxim = 1`.
- Actualizezi `lung` la fiecare pas.
- Actualizezi `maxim`.
- Scrii `maxim`.

Soluția comentată (secvmax.cs)

BUCATA 1 DIN 3

```
using System;
using System.IO;

class Program
{
    static void Main()
    {
        char[] sep = { ' ', '\n', '\r', '\t' }; // separatori: spatiu, rand nou, tab
        string text = File.ReadAllText("secvmax.in");
        string[] t = text.Split(sep, StringSplitOptions.RemoveEmptyEntries);
        int k = 0;
        int n = int.Parse(t[k++]);
        int[] v = new int[n];
        for (int i = 0; i < n; i++)
            v[i] = int.Parse(t[k++]);
    }
}
```

1 Șablonul pentru fișiere.

BUCATA 2 DIN 3

```
int lung = 1, maxim = 1; // seria curenta si cea mai lunga
for (int i = 1; i < n; i++)
{
    if (v[i] == v[i - 1]) lung++; // seria continua
    else lung = 1; // incepe o serie noua
    if (lung > maxim) maxim = lung;
}
```

2 La fiecare pas, seria curentă fie crește, fie se rupe. Maximul îl actualizăm după fiecare pas, ca să nu pierdem seria de la finalul vectorului.

BUCATA 3 DIN 3

```
StreamWriter fout = new StreamWriter("secvmax.out");
fout.WriteLine(maxim);
fout.Close();
}
```

3 Scriem rezultatul și închidem fișierul.

Atenție la:

- Când seria se rupe, `lung` devine 1, nu 0.
- Verifică maximul la fiecare pas, nu doar când se rupe seria.

Întrebări

51. Cât e răspunsul pentru `4 4 1 1 1 1 4`?

- A. 2
- B. 3
- C. 4
- D. 7

52. De ce **lung** devine 1 (și nu 0) când seria se rupe?

- A. Din greșeală
- B. Pentru că elementul curent e deja o serie de lungime 1
- C. Ca să meargă mai repede
- D. Nu contează

6.6 Interclasare (doi sau trei indici)

Unești șiruri ordonate, folosind câte un «deget» pentru fiecare șir.

De pe pbinfo: Interclasare1 (#250), Interclasare3 (#284), comun (#2668). Exerciții suplimentare: Interclasare.

Interclasare1

Unești două șiruri ordonate, fără dubluri

Enunț rezumat. Textul complet: pbinfo.ro/probleme/250/interclasare1

Enunțul pe scurt

În `interclasare1.in` ai un șir `a` cu `n` numere, apoi un șir `b` cu `m` numere. Ambele sunt **strict crescătoare**.

Scrii în `interclasare1.out` toate valorile din ambele șiruri, **crescător**, fiecare valoare **o singură dată**.

Pe fiecare rând pui câte **10** valori (ultimul rând poate avea mai puține).

Comparație. Două cozi de oameni, fiecare ordonată după înălțime. Vrei o singură coadă ordonată. Te uiți doar la primul om din fiecare coadă și îl lași să treacă pe cel mai scund. Dacă sunt la fel de înalți (aceeași valoare), trece doar unul, iar celălalt pleacă acasă.

PRIMEȘTI

`interclasare1.in`: `n`, apoi `n` numere; `m`, apoi `m` numere.

AFIȘEZI

`interclasare1.out`: reuniunea, crescător, câte 10 pe rând.

Exemplu

INTERCLASARE1.IN

```
6
1 3 5 7 9 11
7
2 3 4 8 9 10 12
```

INTERCLASARE1.OUT

```
1 2 3 4 5 7 8 9 10 11
12
```

De ce: 3 și 9 apar în ambele șiruri, dar le scriem o singură dată. Sunt 11 valori: 10 pe primul rând, una pe al doilea.

Exemplul de mai sus e făcut pentru ghid și verificat cu soluția; exemplul oficial e în enunțul de pe pbinfo.

Alte teste: `2 / 1 2 / 2 / 1 2` → `1 2`

Indicii (citește-le pe rând)

- Folosești doi indici: `x` pentru `a` și `y` pentru `b`. Amândoi pornesc de la 0.
- Cât timp ambele șiruri mai au elemente: scrii valoarea mai mică și avansezi doar indicele ei. Dacă sunt egale, scrii o dată și avansezi **ambii** indici.
- La final, unul dintre șiruri mai are elemente: le scrii pe toate. Pentru «10 pe rând» ții un contor și faci `WriteLine()` când ajunge la 10.

Ideea, pe pași

- Funcția `Scrie(x)`, care se ocupă de «10 pe rând».
- Citești `a` (`n` numere) și `b` (`m` numere).

3. Interclasare cu x și y.
4. Scrii ce a rămas din a, apoi din b.
5. Închizi fișierul.

Soluția comentată (interclasare1.cs)

BUCATA 1 DIN 5

```
using System;
using System.IO;

class Program
{
    static StreamWriter fout;
    static int pePeLinie = 0;
```

- 1 Variabile `static` puse în afara lui `Main`, ca să le poată folosi și funcția `Scrie`. `pePeLinie` numără câte valori am scris pe rândul curent.

BUCATA 2 DIN 5

```
// afiseaza x si trece la rand nou dupa fiecare 10 valori
static void Scrie(int x)
{
    fout.Write(x + " ");
    pePeLinie++;
    if (pePeLinie == 10)
    {
        fout.WriteLine();
        pePeLinie = 0;
    }
}
```

- 2 O funcție care scrie un număr și «ține socoteala». La a 10-a valoare trece la rând nou și resetează contorul. `void` = funcția nu întoarce nimic, doar face ceva.

BUCATA 3 DIN 5

```
static void Main()
{
    char[] sep = { ' ', '\n', '\r', '\t' }; // separatori: spatiu, rand nou, tab
    string text = File.ReadAllText("interclasare1.in");
    string[] t = text.Split(sep, StringSplitOptions.RemoveEmptyEntries);
    int k = 0;
    int n = int.Parse(t[k++]);
    int[] a = new int[n];
    for (int i = 0; i < n; i++) a[i] = int.Parse(t[k++]);
    int m = int.Parse(t[k++]);
    int[] b = new int[m];
    for (int i = 0; i < m; i++) b[i] = int.Parse(t[k++]);

    fout = new StreamWriter("interclasare1.out");
```

- 3 Ordinea din fișier: n, șirul a, apoi m, șirul b. Degetul `k` merge mai departe prin toate bucățile.

BUCATA 4 DIN 5

```

int x = 0, y = 0; // x merge prin a, y merge prin b
while (x < n && y < m)
{
    if (a[x] < b[y]) { Scribe(a[x]); x++; }
    else if (a[x] > b[y]) { Scribe(b[y]); y++; }
    else { Scribe(a[x]); x++; y++; } // egale -> o singura data
}

```

- 4 Inima interclasării. Comparăm doar «capetele» celor două cozi. Cel mai mic e scris și coada lui avansează. Dacă sunt egale, scriem o dată și avansăm în ambele.

BUCATA 5 DIN 5

```

while (x < n) { Scribe(a[x]); x++; } // ce a ramas in a
while (y < m) { Scribe(b[y]); y++; } // ce a ramas in b

fout.Close();
}
}

```

- 5 Când un șir s-a terminat, celălalt mai poate avea elemente, toate mai mari, deci le scriem direct. Doar una dintre cele două bucle chiar face ceva.

La încercarea ta: 100 de puncte.

Atenție la:

- Nu uita să scrii ce a rămas după bucla principală.
- La valori egale avansezi ambii indici.
- Rând nou după fiecare 10 valori, nu după fiecare valoare.

Întrebări

53. $a = 1\ 4$, $b = 2\ 4\ 5$. Ce se scrie?

- A. 1 2 4 4 5
- B. 1 2 4 5
- C. 1 4 2 4 5
- D. 1 2 5

54. Când se oprește `while (x < n && y < m)`?

- A. Când ambele șiruri s-au terminat
- B. Când măcar unul dintre șiruri s-a terminat
- C. După 10 valori
- D. Niciodată

Interclasare3

Valorile pare din două șiruri, unul dintre ele invers

Enunț rezumat. Textul complet: pbinfo.ro/probleme/284/interclasare3

Enunțul pe scurt

În `interclasare3.in` ai `n` și `m`, apoi șirul `a` (**crescător**, `n` numere) și șirul `b` (**descrescător**, `m` numere).

Scrii în `interclasare3.out`, crescător, doar valorile **pare** din ambele șiruri, câte **20** pe rând.

Comparație. E interclasarea obișnuită, cu două trucuri: o coadă e întoarsă (o parcurgi de la capăt), iar pe cei cu număr impar îi sari.

PRIMEȘTI

`interclasare3.in`: `n` și `m`, apoi `a`, apoi `b`.

AFIȘEZI

`interclasare3.out`: valorile pare, crescător, câte 20 pe rând.

Exemplu

INTERCLASARE3.IN

```
5 4
1 2 4 7 8
10 6 5 2
```

INTERCLASARE3.OUT

```
2 2 4 6 8 10
```

De ce: Pare din `a`: 2 4 8. Pare din `b`, citit de la coadă: 2 6 10. Unite crescător: 2 2 4 6 8 10 (valoarea 2 apare în ambele, deci de două ori).

Exemplul de mai sus e făcut pentru ghid și verificat cu soluția; exemplul oficial e în enunțul de pe pbinfo.

Alte teste: `3 3 / 2 4 6 / 7 5 3` → `2 4 6`

Indicii (citește-le pe rând)

- Șirul `b` e descrescător, dar dacă îl citești **de la coadă** (`y = m - 1`, apoi `y--`) devine crescător!
- Numerele impare le sari: dacă `a[x]` e impar, `x++` și `continue`. La fel pentru `b[y]`.
- Restul e interclasare obișnuită: scrii valoarea mai mică. La final scrii ce a rămas (tot doar pare).

Ideea, pe pași

- `Scrie(x)` cu 20 pe rând.
- `x` pornește de la 0 în `a`; `y` pornește de la `m - 1` în `b`.
- Sari imparele, interclasezi parele.
- Scrii resturile (doar pare).

Soluția comentată (interclasare3.cs)

BUCATA 1 DIN 4

```
using System;
using System.IO;

class Program
{
    static StreamWriter fout;
    static int pePeLinie = 0;

    static void Scrie(int x)                // 20 de valori pe linie
    {
        fout.Write(x + " ");
        pePeLinie++;
        if (pePeLinie == 20)
        {
            fout.WriteLine();
            pePeLinie = 0;
        }
    }
}
```

- 1 Aceeași funcție ca la Interclasare1, dar cu 20 de valori pe rând.

BUCATA 2 DIN 4

```
static void Main()
{
    char[] sep = { ' ', '\n', '\r', '\t' }; // separatori: spatiu, rand nou, tab
    string text = File.ReadAllText("interclasare3.in");
    string[] t = text.Split(sep, StringSplitOptions.RemoveEmptyEntries);
    int k = 0;
    int n = int.Parse(t[k++]);
    int m = int.Parse(t[k++]);
    int[] a = new int[n];
    int[] b = new int[m];
    for (int i = 0; i < n; i++) a[i] = int.Parse(t[k++]);
    for (int i = 0; i < m; i++) b[i] = int.Parse(t[k++]);

    fout = new StreamWriter("interclasare3.out");
}
```

- 2 Aici n și m sunt amândouă la început, apoi vin șirurile.

BUCATA 3 DIN 4

```
int x = 0;                // a e CRESCATOR -> il parcurgem de la inceput
// b e DESCRESCATOR -> il parcurgem de la COADA (asa devine crescator)
int y = m - 1;
while (x < n && y >= 0)
{
    if (a[x] % 2 != 0) { x++; continue; } // sarim peste impare
    if (b[y] % 2 != 0) { y--; continue; }
    if (a[x] <= b[y]) { Scrie(a[x]); x++; }
    else                { Scrie(b[y]); y--; }
}
```

- 3 **continue** sare direct la următoarea verificare a lui **while**, fără să mai execute restul. **y** scade (**y--**), pentru că mergem de la coada lui b spre început.

```

while (x < n) { if (a[x] % 2 == 0) Scrie(a[x]); x++; }
while (y >= 0) { if (b[y] % 2 == 0) Scrie(b[y]); y--; }

fout.Close();
}
}

```

4 Resturile, tot cu filtrul de numere pare.

La încercarea ta: 100 de puncte.

Atenție la:

- b se parcurge de la `m - 1` în jos (`y--`), iar condiția devine `y >= 0`.
- Filtrul de numere pare se aplică și la resturi.

Întrebări

55. b = 9 6 4 1 (descrescător). În ce ordine îl parcurge `y = m - 1; y--` ?

- A. 9 6 4 1
- B. 1 4 6 9
- C. 6 4
- D. 9 1

56. Ce face `continue` într-un while?

- A. Iese din buclă
- B. Sare la următoarea repetare a buclei
- C. Oprește programul
- D. Repetă instrucțiunea curentă

comun

Cel mai mic număr comun din trei șiruri

Enunț rezumat. Textul complet: pbinfo.ro/probleme/2668/comun

Enunțul pe scurt

Primești n , apoi trei șiruri a , b , c cu câte n numere, fiecare ordonat **crescător**.

Afișezi **cel mai mic** număr care apare în **toate trei** șirurile. Dacă nu există, afișezi **-1**.

Comparație. Trei prieteni au fiecare câte o listă ordonată cu zilele în care sunt liberi. Căutați prima zi în care sunteți toți liberi. Cel care are ziua cea mai mică pe listă trece la următoarea lui zi liberă, până când toți trei arată aceeași zi.

PRIMEȘTI

n , apoi șirurile a , b , c .

AFIȘEZI

Cel mai mic număr comun sau **-1**.

Exemplu

INTRARE

4
1 3 5 7
2 3 5 8
0 5 7 9

IEȘIRE

5

De ce: 3 e în a și b , dar nu în c . 5 e în toate trei.

Exemplul de mai sus e făcut pentru ghid și verificat cu soluția; exemplul oficial e în enunțul de pe pbinfo.

Alte teste: 2 / 1 2 / 3 4 / 5 6 → -1

Indicii (citește-le pe rând)

1. Trei indici, câte unul pe fiecare șir: i_1 , i_2 , i_3 .
2. Dacă toate trei valorile sunt egale, ai răspunsul. Și e cel mai mic posibil, pentru că șirurile sunt crescătoare.
3. Altfel, avansezi indicele care arată valoarea **cea mai mică**: ea sigur nu poate fi comună, pentru că celelalte șiruri au trecut deja de ea.

Ideea, pe pași

1. Citești cele trei șiruri.
2. Trei indici la 0.
3. Egale → gata. Altfel avansezi indicele cu minimul.
4. Afișezi răspunsul sau -1.

BUCATA 1 DIN 3

```
using System;

class Program
{
    static void Main()
    {
        char[] sep = { ' ', '\n', '\r', '\t' }; // separatori: spatiu, rand nou, tab
        string[] t = Console.In.ReadToEnd().Split(sep, StringSplitOptions.RemoveEmptyEntries);
        int k = 0;
        int n = int.Parse(t[k++]);
        int[] a = new int[n], b = new int[n], c = new int[n];
        for (int i = 0; i < n; i++) a[i] = int.Parse(t[k++]);
        for (int i = 0; i < n; i++) b[i] = int.Parse(t[k++]);
        for (int i = 0; i < n; i++) c[i] = int.Parse(t[k++]);
    }
}
```

- 1 Trei vectori declarați pe o singură linie, apoi citiți unul după altul.

BUCATA 2 DIN 3

```
// trei "degete", cate unul pe fiecare sir;
// avansam mereu degetul cu valoarea cea mai mica
int i1 = 0, i2 = 0, i3 = 0;
int rasp = -1;
while (i1 < n && i2 < n && i3 < n)
{
    if (a[i1] == b[i2] && b[i2] == c[i3])
    {
        // prima gasita = cea mai mica (sirurile sunt crescatoare)
        rasp = a[i1];
        break;
    }
}
```

- 2 Dacă toate trei degetele arată aceeași valoare, am găsit-o. `rasp = -1` de la început rezolvă singur cazul «nu există».

BUCATA 3 DIN 3

```
int mn = Math.Min(a[i1], Math.Min(b[i2], c[i3]));
if (a[i1] == mn) i1++;
else if (b[i2] == mn) i2++;
else i3++;
}

Console.WriteLine(rasp);
}
```

- 3 `Math.Min(x, y)` dă minimul dintre două numere; pus de două ori, dă minimul dintre trei. Degetul care arată minimul avansează.

La încercarea ta: 100 de puncte.

Atenție la:

- Folosește `&&` între cele trei condiții de oprire.
- Avansezi degetul cu valoarea minimă, nu pe toate.

Întrebări

57. $a[i1] = 4$, $b[i2] = 7$, $c[i3] = 5$. Ce indice avansează?

- A. $i1$
- B. $i2$
- C. $i3$
- D. toți

58. Cât dă `Math.Min(8, Math.Min(3, 6))` ?

- A. 8
- B. 6
- C. 3
- D. 17

Interclasare

Interclasarea simplă a doi vectori

Făcut pentru antrenament, în stilul problemelor de pe pbinfo. Soluția e verificată pe toate testele.

Enunțul pe scurt

Primești un vector **a** cu **n** numere și un vector **b** cu **m** numere, ambele ordonate **crescător**.

Afișezi, pe un rând, toate cele $n + m$ numere, în ordine **crescătoare**. Valorile care apar în ambii vectori se scriu de fiecare dată.

Comparație. Ai două teancuri de foi ordonate după număr. Te uiți la foaia de sus din fiecare teanc și o iei mereu pe cea cu numărul mai mic.

PRIMEȘTI

n, apoi a; m, apoi b.

AFIȘEZI

Toate valorile, crescător, pe un rând.

Exemplu

INTRARE

4
1 4 6 9
3
2 4 10

IEȘIRE

1 2 4 4 6 9 10

Alte teste: 2 / 5 6 / 2 / 1 2 → 1 2 5 6

Indicii (citește-le pe rând)

- Doi indici: **x** pentru **a**, **y** pentru **b**, amândoi de la 0.
- Cât timp ambii vectori mai au elemente: scrii valoarea mai mică (**<=** ca să meargă și la egalitate) și avansezi doar indicele ei.
- La final scrii ce a rămas în **a**, apoi ce a rămas în **b**.

Ideea, pe pași

- Citești a și b.
- Interclasare cu x și y.
- Scrii resturile.
- Rând nou la final.

BUCATA 1 DIN 2

```
using System;

class Program
{
    static void Main()
    {
        char[] sep = { ' ', '\n', '\r', '\t' }; // separatori: spatiu, rand nou, tab
        string[] t = Console.In.ReadToEnd().Split(sep, StringSplitOptions.RemoveEmptyEntries);
        int k = 0;
        int n = int.Parse(t[k++]);
        int[] a = new int[n];
        for (int i = 0; i < n; i++) a[i] = int.Parse(t[k++]);
        int m = int.Parse(t[k++]);
        int[] b = new int[m];
        for (int i = 0; i < m; i++) b[i] = int.Parse(t[k++]);
    }
}
```

- 1 Citim n și a, apoi m și b.

BUCATA 2 DIN 2

```
int x = 0, y = 0;
while (x < n && y < m)
{
    if (a[x] <= b[y]) { Console.Write(a[x] + " "); x++; }
    else { Console.Write(b[y] + " "); y++; }
}
while (x < n) { Console.Write(a[x] + " "); x++; } // restul din a
while (y < m) { Console.Write(b[y] + " "); y++; } // restul din b
Console.WriteLine();
}
```

- 2 Varianta de bază a interclasării. Spre deosebire de Interclasare1, aici valorile egale se scriu de două ori, deci nu e nevoie de cazul separat «egale».

Atenție la:

- Nu uita resturile după bucla principală.
- Toate valorile pe același rând, deci `Write`.

Întrebări

59. $a = 2\ 5$, $b = 1\ 5\ 7$. Ce se afișează?

- A. 1 2 5 7
- B. 1 2 5 5 7
- C. 2 5 1 5 7
- D. 1 5 7 2 5

60. Câte valori se afișează în total?

- A. n
- B. m
- C. $n + m$
- D. $n \cdot m$

7. Dictionar

/ (împărțire întreagă): Între două int, împărțirea taie zecimalele. `17 / 5 == 3`

% (restul): Restul împărțirii. $x \% 2 == 0$ înseamnă par, $x \% 10$ e ultima cifră. `17 \% 5 == 2`

bool (steguleț): Poate fi doar true (adevărat) sau false (fals). Perfect pentru «am găsit?», «e prim?». `bool gasit = false;`

break: Leși imediat din bucla în care ești (doar din cea mai apropiată). `if (x == 0) break;`

Căutare: Parcurgi vectorul și verifici dacă o valoare apare. Folosești un bool «gasit». `if (v[i] == x) gasit = true;`

Cifrele unui număr: Ultima cifră e $x \% 10$; o tai cu $x /= 10$. Repeti cât timp $x > 0$. `472 → 2, 7, 4`

Citare (șablonul): Citești tot inputul o dată, îl tai în bucăți și iei numerele pe rând cu $t[k++]$. `int n = int.Parse(t[k++]);`

cmmdc: Cel mai mare divizor comun a două numere. Se calculează cu algoritmul lui Euclid (resturi repetate). `cmmdc(18, 12) = 6`

continue: Sari peste restul buclei și treci direct la următoarea repetare. `if (x \% 2 != 0) continue;`

Contor: O variabilă care numără. Pornește de la 0 și crește cu $cnt++$ de fiecare dată când găsești ceva. `int cnt = 0; ... cnt++;`

Divizor: d e divizor al lui x dacă x se împarte exact la d, adică $x \% d == 0$. `divizorii lui 12: 1, 2, 3, 4, 6, 12`

Eroare de compilare (E.C.): Programul nu poate nici măcar porni: o paranteză lipsă, un ; uitat, un nume scris greșit. Primești 0 puncte. `lipsește ) sau ;`

Fișier .in / .out: Unele probleme citesc dintr-un fișier (.in) și scriu în altul (.out) în loc de tastatură și ecran. `File.ReadAllText("minmax.in")`

for: Buclă pentru când știi de câte ori repeți: de unde pornești; cât timp continui; cum avansezi. `for (int i = 0; i < n; i++)`

Funcție: O bucată de cod cu nume, pe care o scrii o dată și o folosești de mai multe ori. Primește valori și poate da înapoi un rezultat cu return. `static int SumaCifre(int x) { ... }`

Indice (poziție): Numărul cutiei dintr-un vector. În C# pornește de la 0: primul element e $v[0]$, ultimul $v[n - 1]$. `v[0], v[1], ..., v[n-1]`

int: Tipul pentru numere întregi obișnuite (cam $\pm 2,1$ miliarde). `int n = 10;`

Interclasare: Unești două șiruri ordonate într-unul ordonat, folosind câte un indice pentru fiecare și luând mereu valoarea mai mică. `1 4 + 2 3 → 1 2 3 4`

long: Tipul pentru numere întregi foarte mari. Îl folosești când un calcul poate depăși int. `long og = 9876543210;`

Maxim: Cea mai mare valoare. Pornești de la primul element și compari restul cu ea. `mx = v[0];`

Minim: Cea mai mică valoare. Tot de la primul element pornești. `mn = v[0];`

Multiplu: Un multiplu al lui k e k înmulțit cu un număr întreg. `multiplii lui 3: 0, 3, 6, 9, 12...`

Număr prim: Are exact doi divizori: 1 și el însuși. 0 și 1 nu sunt prime; 2 e singurul prim par. `2, 3, 5, 7, 11, 13`

Oglindit: Numărul citit de la coadă la cap. Se construiește cu $r = r * 10 + x \% 10$. `ogluitul lui 123 este 321`

Palindrom: Ceva ce se citește la fel din ambele părți. `12321, «cojoc»`

Parcursere: Treci prin toate elementele unui vector, unul câte unul, de obicei cu un for. `for (int i = 0; i < n; i++)`

Parcursere inversă: Treci prin vector de la coadă spre început. `for (int i = n - 1; i >= 0; i--)`

Pătrat perfect: Un număr care e alt număr întreg înmulțit cu el însuși. `16 = 4·4, 49 = 7·7`

Pereche (i, j): Două poziții diferite. Ca să nu numeri aceeași pereche de două ori, j pornește de la $i + 1$. `for i ... for (int j = i + 1; ...)`

Prime între ele: Două numere al căror cmmdc e 1. NU trebuie să fie ele însele prime! `8 și 9 sunt prime între ele`

Produs pe cruce: $a/b == c/d$ e același lucru cu $a·d == b·c$. Așa compari fracții fără împărțiri cu virgulă. `2/3 == 4/6 pentru că 2·6 == 3·4`

Secvență: O bucată de elemente vecine dintr-un vector (fără găuri). `în 4 7 1 9, «7 1» e o secvență, «4 1» nu`

Sortare: Ordonarea elementelor. În C#: `Array.Sort(v)` ordonează crescător. `Array.Sort(v);`

Sumă (acumulare): Pornești de la 0 și aduni fiecare valoare: `s += x`. `int s = 0; s += v[i];`

Șablon: Codul de pornire, mereu același, pe care îl copiezi la fiecare problemă.

Ștergere din vector: Păstrezi doar elementele bune, mutându-le în față cu un al doilea indice m. La final `n = m`.
`v[m] = v[i]; m++;`

Variabilă: O cutie cu nume în care ții o valoare. Valoarea se poate schimba. `int x = 5;`

Vector: Un rând de cutii lipite, cu un singur nume. Fiecare cutie are un număr de ordine (indice). `int[] v = new int[n];`

Vector de frecvență: Un vector în care `fr[x]` spune de câte ori a apărut valoarea x. Folosești numărul citit ca poziție.
`fr[x]++;`

while: Buclă care se repetă cât timp o condiție e adevărată.
`while (x > 0) { ... }`

Write / WriteLine: `Write` afișează și rămâne pe rând. `WriteLine` afișează și trece la rândul următor.
`Console.Write(x + " ");`

8. Răspunsurile la întrebări

Încearcă întâi singur(ă). Explicația contează mai mult decât litera.

1. B (0). Niciun număr nu e mai mare decât 0, deci `max` rămâne 0, deși 0 nu e printre numere. De-asta primul număr devine direct maxim. `Maxim`

2. B (Iese din buclă și continuă cu ce e după ea). `break` doar te scoate din buclă. Programul continuă cu instrucțiunile de după `}`-ul buclei. `Maxim`

3. B (Dacă toate numerele sunt mai mari de 10000, minimul iese 10000, care nici nu e în listă). Cu numere de până la 1.000.000, toate pot fi peste 10000. Pornind de la `v[0]`, răspunsul e mereu un număr din listă. `MinMax`

4. C (Fișierul de ieșire poate rămâne gol). Textul stă întâi într-o memorie temporară. `Close()` îl scrie efectiv pe disc. `MinMax`

5. B (2). Se păstrează 5 și 7, deci `m` ajunge la 2. `stergere1`

6. B (`v[i]` e impar). Restul împărțirii la 2 e diferit de 0, deci numărul e impar. `stergere1`

7. B (7). 8 e par, deci cel mai mare impar ≤ 8 este 7. `Afisare`

8. B (2 4 6). i ia valorile 2, 4, 6. La 8 condiția `i <= 7` nu mai e adevărată. `Afisare`

9. B (Adună `v[i]` la suma). `+=` adună la valoarea care era deja în variabilă. `SumaPare`

10. C (6). $2 + 4 = 6$. `SumaPare`

11. B (3). Cu `>=`, al doilea 8 îl înlocuiește pe primul, deci `poz = 2` (indice), adică poziția 3. `PozMax`

12. B (Indicii C# pornesc de la 0, enunțul numără de la 1). Primul element e `v[0]`, dar în enunț e «poziția 1». `PozMax`

13. A (16). $9 + 0 + 7 = 16$. `Numarare5`

14. B (6). (0,1) (0,2) (0,3) (1,2) (1,3) (2,3) = 6 perechi. `Numarare5`

15. A (Da, `cmmdc = 1`). Divizorii lui 8: 1, 2, 4, 8. Ai lui 15: 1, 3, 5, 15. Singurul comun e 1. `Numarare4`

16. C (6). $18 \% 12 = 6$, apoi $12 \% 6 = 0$. Răspuns: 6. `Numarare4`

17. B (504). Cifrele de la coadă sunt 0, 5, 0, 4, adică 0504, care ca număr e 504. `EgaleOglindit`

18. B (`r = r * 10 + x % 10`). Înmulțim cu 10 ca să facem loc și adăugăm ultima cifră. `EgaleOglindit`

19. A (98). $98765 \rightarrow 9876 \rightarrow 987 \rightarrow 98$ ($98 < 100$, ne oprim). `ElemPP`

20. B (`fout.Write(x + " ")`). `Write` rămâne pe rând, iar spațiul separă numerele. `ElemPP`

21. B (7). $7 \cdot 7 = 49 \leq 49$, deci se verifică și `d = 7`, iar $49 \% 7 == 0$, deci nu e prim. `NrApPrime`

22. C (29). $21 = 3 \cdot 7$, $27 = 3 \cdot 9$, iar 1 nu e prim. 29 nu se împarte decât la 1 și 29. `NrApPrime`

23. C (3). 5 apare de trei ori. `Prime4`

24. B (Ar câștiga cel mai mare, adică greșit). Cu `>=`, un număr mai mare, găsit mai târziu, l-ar înlocui pe cel mic. `Prime4`

25. B (15). `r = 3`. Până la 10 sunt 3 pași, până la 15 sunt 2 pași, deci 15. `inlocuireK`

26. A (Pentru că `ReadLine` citește tot rândul "3 5", care nu e un singur număr). `int.Parse("3 5")` dă eroare: textul conține un spațiu și două numere. `inlocuireK`

27. A (19). $4 + 0 + 9 + 6 = 19$. `SumaCifreMax`

28. A (19). Toate au suma 10, deci rămâne primul: 19. `SumaCifreMax`

29. **B** (1001). 1001 citit de la coadă e tot 1001. **Palindroame**

30. **B** (Pentru că oglinditul unui număr mare poate depăși int). De exemplu, oglinditul lui 1.000.000.009 e 9.000.000.001, peste limita lui int. **Palindroame**

31. **B** (Programul crapă (IndexOutOfRangeException)). Locurile sunt v[0], v[1], v[2]. v[3] nu există. **VerifDiferite**

32. **B** (= pune o valoare, == compară). x = 5 pune 5 în x. x == 5 întreabă «e x egal cu 5?». **VerifDiferite**

33. **B** (-1). În C#, restul are semnul primului număr. De aceea testăm imparul cu != 0. **PareOrdonate**

34. **A** (DA). Parele: 6, 6, 8. Egalitatea e permisă, deci cresc. **PareOrdonate**

35. **B** (4). $n - 1 - i = 6 - 1 - 1 = 4$. **PalXXL**

36. **B** (3). Împărțirea între int-uri taie zecimalele. **PalXXL**

37. **B** (1 4 5). Sortează crescător, în vector. Nu afișează nimic. **Identice**

38. **B** (În acoladele if-ului, după ok = false). Fără acolade, if-ul se aplică doar primei instrucțiuni. **Identice**

39. **A** ($3 \cdot 12 == 4 \cdot 9 \rightarrow 36 == 36 \rightarrow$ DA). Produsul pe cruce: $a \cdot d == b \cdot c$. **Proportionale**

40. **B** (4). $5 / 2 = 2$ (se taie zecimala), iar $2 \cdot 2 = 4$. **Proportionale**

41. **B** (NU). 5 nu e mai mare decât 5. **Crescator**

42. **B** (Se accesează v[-1] și programul crapă). Pentru i = 0, v[i - 1] înseamnă v[-1], care nu există. **Crescator**

43. **B** (30 20 10). Sortat: 10 20 30. De la coadă: 30 20 10. **Sort10**

44. **C** (1 și 2 pe rânduri separate). **WriteLine** trece la rând nou după fiecare afișare. **Sort10**

45. **B** (2 2). Sortat: 2 2 6 9. Primele două: 2 2. **KMici**

46. **B** (Pentru că k e deja numele unei date din enunț și n-ai voie două variabile cu același nume). Două variabile cu același nume în același loc dau eroare de compilare. **KMici**

47. **B** (3). [5 6] [2 2 9] [1], adică scade de două ori, deci $1 + 2 = 3$. **Secvente**

48. **B** (Pentru că îl comparăm cu v[i - 1], iar v[-1] nu există). Pentru i = 0, v[i - 1] ar fi v[-1], deci programul ar crăpa. **Secvente**

49. **A** (1 3). De la poziția 1: 1, 2, 5, deci pozițiile 1...3. E cea mai din stânga. **SumaInSecv**

50. **B** (oprește doar for-ul interior). **break** iese doar din bucla cea mai apropiată. **SumaInSecv**

51. **C** (4). Seria de patru de 1. **SecvMax**

52. **B** (Pentru că elementul curent e deja o serie de lungime 1). Elementul nou începe singur o serie, deci are lungimea 1. **SecvMax**

53. **B** (1 2 4 5). 4 apare în ambele, dar se scrie o singură dată. **Interclasare1**

54. **B** (Când măcar unul dintre șiruri s-a terminat). && cere ca ambele condiții să fie adevărate. Când una devine falsă, bucla se oprește. **Interclasare1**

55. **B** (1 4 6 9). Pornim de la ultima poziție (1) spre prima (9), deci crescător. **Interclasare3**

56. **B** (Sare la următoarea repetare a buclei). Restul corpului buclei e sărit, iar condiția e verificată din nou. **Interclasare3**

57. **A** (i1). Minimul e 4, în a, deci avansează i1. **comun**

58. **C** (3). Minimul dintre 3 și 6 e 3, apoi minimul dintre 8 și 3 e 3. **comun**

59. B (1 2 5 5 7). Valoarea 5 apare în ambii vectori, deci se scrie de două ori. [Interclasare](#)

60. C ($n + m$). Fiecare element din a și din b ajunge o dată în rezultat. [Interclasare](#)