Probleme consultații
Șiruri (tablouri unidimensionale)

Problema 1 - Riffle Shuffle

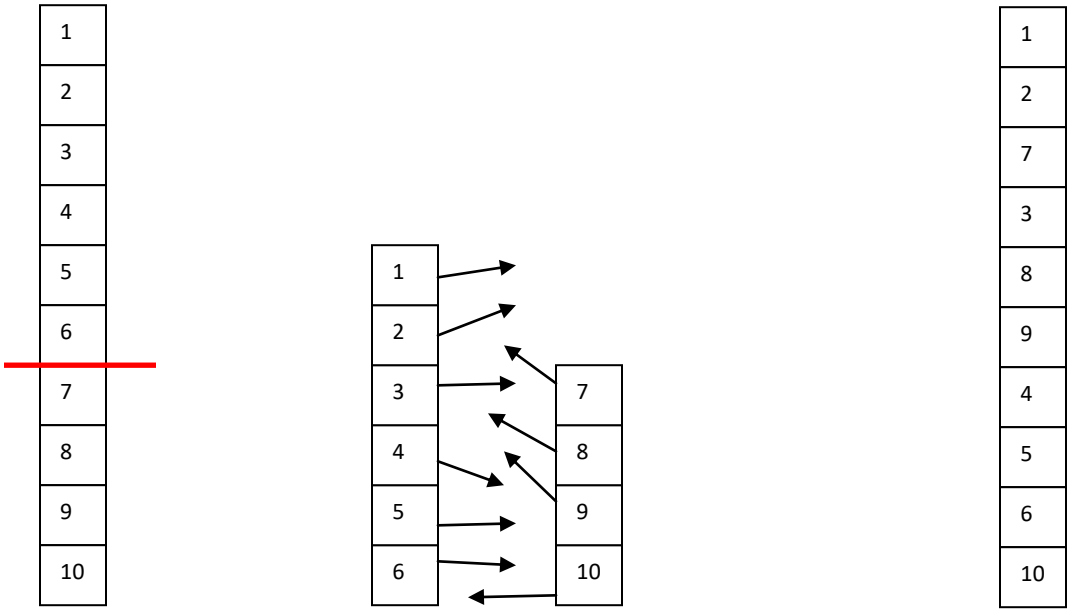
Enunț

Avem un pachet de *n* cărți; pe fiecare carte avem un număr unic de la 1 la *n*. Procesul de amestecare a cărților, numit ”riffle shuffle”, împarte pachetul aleatoriu în 2 jumătăți și intercalează jumătățile în mod aleatoriu (vezi imaginea din dreapta). Având 2 ordonări ale cărților, ordonare_inițială și ordonare_finală, determinați dacă pornind de la ordonare_inițială putem ajunge, cu un singur riffle shuffle, la ordonare_finală.

Date de intrare sunt *n* (numărul de cărți), *ordonare inițială* (un tablou cu n elemente) și *ordonare finală* (un tablou cu n elemente). Rezultatul este *adevărat* sau *fals*.

De exemplu, dacă avem 10 cărți, ordonare_inițială este [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] și ordonare_finală este [1, 2, 7, 3, 8, 9, 4, 5, 6, 10], se poate ajunge de la ordonarea inițială la cea finală cu un singur riffle shuffle (vedeți exemplul de mai jos). Pornind de la aceeași ordonare_inițială, nu se poate ajunge la ordonare_finală [1, 7, 3, 4, 2, 8, 9, 5, 6, 10].

Pachetul inițial: **Procesul de ”riffle shuffle”** **Pachetul final**



Analiză

Rezolvarea problemei este compusă din 2 părți:
- determinăm punctul unde a fost tăiat pachetul (pe baza ordonării finale);
- având cele 2 jumătăți de pachet (una de la început până la punctul de tăiere, iar a doua de la punctul de tăiere până la capăt), verificăm dacă se poate ajunge la ordonarea finală.

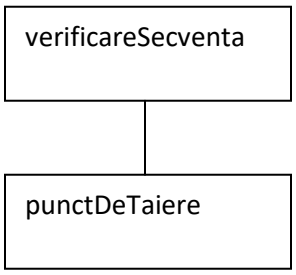
Determinarea punctului de tăiere

Știm că dacă tăiem pachetul de cărți la un punct p, vom avea 2 jumătăți: prima jumătate începe la poziția 1 și se termină la poziția p; 2-a jumătate începe la poziția p+1 și se termină la poziția n. Dacă urmărim exemplul, observăm că ultimul element din ordonarea finală este ultimul element dintr-una dintre jumătăți.
Dacă ultimul element din ordonarea finală nu este egal cu ultimul element din ordonarea inițială, el trebuie să fie ultimul element din prima jumătate. Cum numerele de pe cărți sunt unice, putem găsi imediat punctul de tăiere. Dacă ultimul element din ordonarea finală este egal cu ultimul element din ordonarea inițială, ne uităm la penultimul element din ordonarea finală; acesta din urmă trebuie să fie ori ultimul element din prima jumătate, ori penultimul element din a 2-a jumătate etc.

Verificarea secvenței finale

Odată ce avem punctul de tăiere, parcurgem în paralel pachetul final și cele 2 jumătăți. Parcurgerile se fac de la finalul tabloului spre început (pentru că așa se construiește rezultatul amestecării). La fiecare pas verificăm din care jumătate provine elementul curent din ordonarea finală. În fiecare jumătate vom avea un element curent; tot timpul unul dintre elementele curente din cele 2 jumătăți trebuie să fie următorul element din ordonarea finală.

Identificarea subalgoritmilor



Specificarea subalgoritmilor

Funcție **punctDeTaiere(n, ordI, ordF):**
Descriere: caută punctul de tăiere în *ordF*, pornind de la ordinea inițială *ordI*
Date: n ∈ N*, n este numărul de cărți (lungimea tablourilor ordI și ordF)

ordI – tabloul inițial cu cărți, ordI = ($ordI_i$ |i =1…n, $ordI_i$ ϵ {1,…,n})
ordF - tabloul final cu cărți, ordF = ($ordF_i$ |i =1…n, $ordF_i$ ϵ {1,…,n})
Rezultate: returnează poziția de sfârsit pentru prima jumătate (poziția după care ordI a fost tăiat; poziția ϵ 1 … n) sau -1 dacă cele 2 ordonări sunt identice.

Funcție **verificareSecventa(n, ordI, ordF)**:
Descriere: verifică dacă se poate ajunge de la ordI la ordF cu un singur riffle shuffle
Date: n ϵ N*, n este numărul de cărți (lungimea tablourilor ordI și ordF)
ordI – tabloul inițial cu cărți, ordI = ($ordI_i$ |i =1…n, $ordI_i$ ϵ {1,…,n})
ordF - tabloul final cu cărți, ordF = ($ordF_i$ |i =1…n, $ordF_i$ ϵ {1,…,n})
Rezultate: returnează *true* dacă se poate ajunge din ordI la ordF cu un singur riffle shuffle, *false* altfel

Proiectare

Observație: La toate problemele din acest document indexarea tablourilor începe de la:
- 1 la Pseudocod și Pascal
- 0 la C++

```
funcție punctDeTaiere(n, ordI, ordF) este:
        pozitieOrdI = n             {pornim de la capătul tablourilor}
        pozitieOrdF = n
        cat-timp pozitieOrdF > 0 și ordI[pozitieOrdI] = ordF[pozitieOrdF] execută
                pozitieOrdI ← pozitieOrdI - 1
                pozitieOrdF ← pozitieOrdF - 1
        sf-cat-timp
        dacă pozitieOrdF = 0 atunci       {cele 2 tablouri sunt identice, returnăm -1}
                punctDeTaiere ← -1
        altfel {cautam pozitia elementului cu care se incheie prima jumatate in ordI}
                ultimElemJumatate1 ← ordF[pozitieOrdF]
                pozitieUltimElem ← 1
                cat-timp ordI[pozitieUltimElem] != ultimElemJumatate1 execută
                        pozitieUltimElem ← pozitieUltimElem + 1
                sf-cat-timp
                punctDeTaiere ← pozitieUltimElem
        sf_dacă
sf_functie

functie verificareSecventa(n, ordI, ordF) este:
        taietura ← punctDeTaiere(n, ordI, ordF)
        dacă taietura = -1 atunci
                verificareSecventa ← true
        altfel
                {verificam jumatatile de la capat si ordonare finala de la capat}
                pozJ1 ← taietura    {prima jumatate e de la poz 1 la poz taietura}
                pozJ2 ← n           {a 2-a jumatate e de la poz taietura+1 la poz n}
                pozOrdF ← n
                cat-timp pozOrdF > 0 executa
                        dacă pozJ1 > 0 si ordF[pozOrdF] = ordI[pozJ1] atunci
                                pozOrdF ← pozOrdF - 1
                                pozJ1 ← pozJ1 - 1
                        altfel
```

```
                        dacă pozJ2 > taietura si ordF[pozOrdF] = ordI[pozJ2] atunci
                              pozOrdF ← pozOrdF - 1
                              pozJ2 ← pozJ2 - 1
                        altfel
                              verificareSecventa ← false
{elementul din ordF nu se potriveste cu elementul curent din niciuna dintre jumatati}
                        sf_daca
                  sf_daca
            sf-cat-timp
            verificareSecventa ← true
      sf_daca
sf_functie
```

Exemple

Date de intrare			Rezultat
n	ordI	ordF	
10	[1,2,3,4,5,6,7,8,9,10]	[1,2,7,3,8,9,4,5,6,10]	True
10	[1,2,3,4,5,6,7,8,9,10]	[1,4,5,6, 2,3,7,8,9,10]	True
10	[1,2,3,4,5,6,7,8,9,10]	[1,8,2,3,4,5,6,7,9,10]	True
10	[1,2,3,4,5,6,7,8,9,10]	[10,9,8,7,6,5,4,3,2,1]	False
10	[1,2,3,4,5,6,7,8,9,10]	[1,6,2,8,3,4,5,7,9,10]	False
15	[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]	[9,10,1,2, 11,3,5,12,13,4,6,7,14,15,8]	False
15	[3,7,1,8,10,9,15,2,13,14,4,6,11,5,12]	[3, 13,14,7,4,1,8,6,10,11,5,9,12,15,2]	True

Cod sursă C++

```
int punctDeTaiere(int n, int ordI[], int ordF[])

{

	int pozitieOrdF = n - 1;
	int pozitieOrdI = n - 1;

	while (pozitieOrdF >= 0 && ordI[pozitieOrdI] == ordF[pozitieOrdF])
	{
		pozitieOrdF--;
		pozitieOrdI--;
	}

	if (pozitieOrdF == -1)
	{
		//cele 2 tablouri sunt identice, returnam -1
		return -1;
	}
	else
	{
		int ultimElemJumatate1 = ordF[pozitieOrdF];
		int pozitieUltimElem = 0;
		while (ordI[pozitieUltimElem] != ultimElemJumatate1)
```

```
		{
			pozitieUltimElem++;
		}
		return pozitieUltimElem;
	}
}


bool verificareSecventa(int n, int ordI[], int ordF[])
{

	int taietura = punctDeTaiere(n, ordI, ordF);

	if (taietura == -1)
	{
		//cele 2 tablouri sunt identice
		return true;
	}
	else
	{
		int pozJumatate1 = taietura;
		int pozJumatate2 = n-1;
		int pozOrdF = n-1;

		while (pozOrdF >= 0)
		{
			if (pozJumatate1 >= 0 && ordF[pozOrdF] == ordI[pozJumatate1])
			{
				pozOrdF--;
				pozJumatate1--;
			}
			else if (pozJumatate2 > taietura && ordF[pozOrdF] ==
					ordI[pozJumatate2]) {
				pozOrdF--;
				pozJumatate2--;
			}
			else
			{
				return false;
			}
		}
		return true;
	}
}
```

Cod sursă Pascal

```
program Problema1;

type
    myArray = array[1..100] of integer;

function punctDeTaiere(n:integer; ordI: myArray; ordF: myArray): integer;
var
   pozOrdI: integer;
   pozOrdF: integer;
```

```
   result: integer;
   pozUltimElem: integer;
   ultimElemJumatate1: integer;
begin
    pozOrdI := n;
    pozOrdF := n;
    while ((pozOrdF > 0) and (ordI[pozOrdI] = ordF[pozOrdF])) do
    begin
        pozOrdF := pozOrdF - 1;
        pozOrdI := pozOrdI - 1;
    end;
    if pozOrdF = 0 then
        result := -1
    else
        begin
            ultimElemJumatate1 := ordF[pozOrdF];
            pozUltimElem := 1;
            while ordI[pozUltimElem] <> ultimElemJumatate1 do
            begin
                pozUltimElem := pozUltimElem + 1;
            end;
            result := pozUltimElem;
        end;
    punctDeTaiere:= result;
end;

function verificareSecventa(n: integer; ordI: myArray; ordF: myArray): boolean;
var
    result: boolean;
    taietura: integer;
    pozJumatate1: integer;
    pozJumatate2: integer;
    pozOrdF: integer;
    continua: boolean;
begin
    taietura := punctDeTaiere(n, ordI, ordF);
    if taietura = -1 then
        result := true
    else
    begin
        pozJumatate1 := taietura;
        pozJumatate2 := n;
        pozOrdF := n;
        continua := true;
        while ((continua = true) and (pozOrdF > 0)) do
        begin
            if ((pozJumatate1 > 0) and (ordI[pozJumatate1] = ordF[pozOrdF])) then
            begin
                pozJumatate1 := pozJumatate1 - 1;
                pozOrdF := pozOrdF - 1;
            end
            else if ((pozJumatate2>taietura) and (ordI[pozJumatate2]=ordF[pozOrdF])) then
            begin
                pozJumatate2 := pozJumatate2 - 1;
                pozOrdF := pozOrdF - 1;
            end
            else
                continua := false;
        end;
```

```
        result:= continua;
    end;
    verificareSecventa := result;
end;
```

Problema 2 - Monitorizarea climei

Enunț

Se cere implementarea unei aplicații care analizează date climatice din 2 județe, J1 și J2. Între altele, aplicația va permite:

1.

- citirea numărului natural $p \in \{1, ..., 17\}$, a unei săptămâni de început s și a unei săptămâni de sfârșit f; $s, f \in \{1, ..., 52\}$, iar $f - s >= 3*p$;

- citirea numerelor reale v_s, v_f, c_s, c_f; v_s, $v_f \in (1, 20]$, $v_s <= v_f$; c_s, $c_f \in [0, 1]$, $c_s <= c_f$;

2. citirea, pentru fiecare săptămână σ din săptămâna s până în săptămâna f inclusiv, a temperaturii medii $TMS_{\sigma,Ji}$ în săptămâna respectivă în două județe, J1 și J2 (i ∈ {1, 2}); $TMS_{\sigma,Ji} \in [-40, 50]$;

3. determinarea și afișarea celei mai lungi perioade de timp în care are loc următorul fenomen în ambele județe:

- o perioadă cu *p c-stagnări** consecutive ale temperaturii medii săptămânale (TMS) este precedată de o perioadă cu *p v-creșteri** consecutive SAU de o perioadă cu *p v-scăderi** consecutive ale TMS;

- o perioadă cu *p v-scăderi* consecutive ale TMS este precedată în ordine cronologică de: o perioadă cu *p v-creșteri* consecutive ale TMS și de o perioadă cu *p c-stagnări* consecutive ale TMS;

- o perioadă cu *p v-creșteri* consecutive ale TMS este precedată în ordine cronologică de: o perioadă cu *p v-scăderi* consecutive ale TMS și de o perioadă cu *p c-stagnări* consecutive ale TMS;

- o secvență îndeplinește proprietatea dacă începe cu *p v-creșteri* consecutive;

* fie s_1 și s_2 două săptămâni consecutive:
- în s_2 are loc o *v-creștere* a TMS față de s_1 dacă $TMS_{s2} - TMS_{s1} \in [v_s, v_f]$;
- în s_2 are loc o *c-stagnare* a TMS față de s_1 dacă $|TMS_{s1} - TMS_{s2}| \in [c_s, c_f]$;
- în s_2 are loc o *v-scădere* a TMS față de s_1 dacă $TMS_{s1} - TMS_{s2} \in [v_s, v_f]$;

- se afișează și numărul de secvențe cu *v-creșteri, c-stagnări, v-scăderi*; începând cu a doua săptămână s_i a perioadei identificate, se afișează și diferența dintre TMS din săptămâna s_i și TMS din săptămâna precedentă s_{i-1};

- cea mai lungă perioadă de timp căutată are cel puțin 3*p+1 săptămâni consecutive: *p v-creșteri* urmate de *p c-stagnări* succedate de *p v-scăderi* au loc într-o perioadă care include 3*p+1 săptămâni consecutive;

- în cazul în care există mai multe soluții - mai multe perioade de timp care au aceeași lungime - se afișează una dintre ele;

4. determinarea și afișarea celei mai lungi perioade de timp în care TMS în județul J1 este mai mare decât TMS înregistrată în județul J2 - **temă**.

Exemplul 1

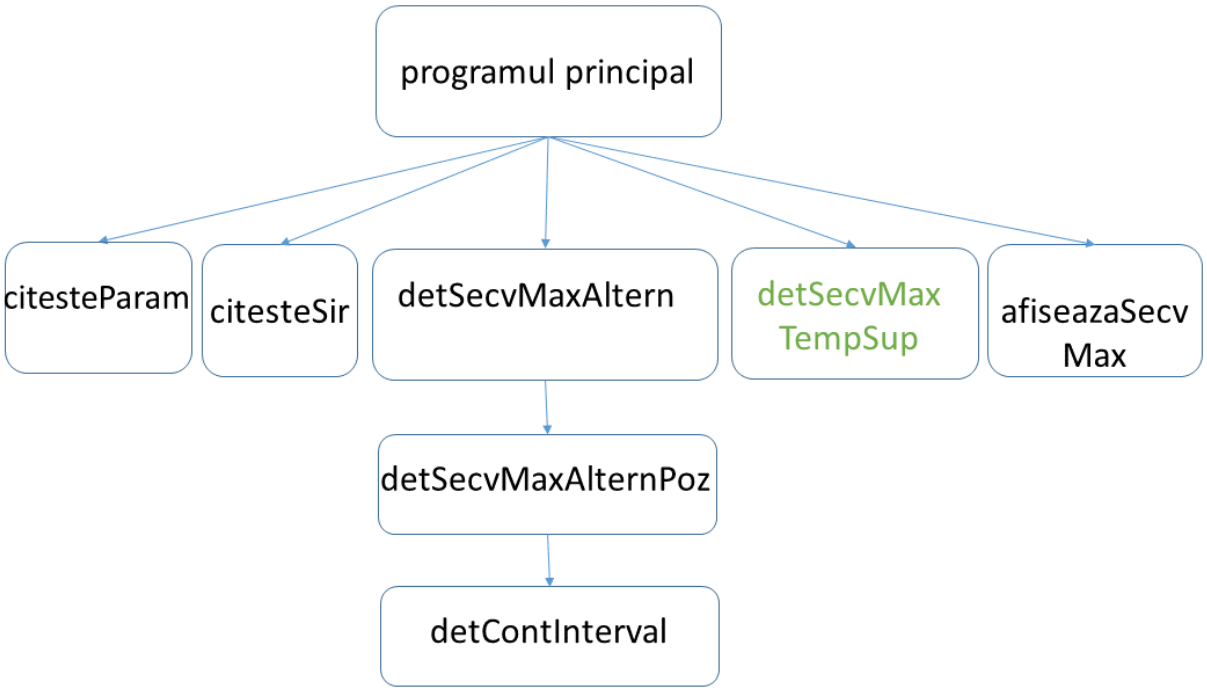
- săptămâna de început s = 2 și săptămâna de sfârșit f = 19
- v_s =3, v_f = 7, c_s = 0, c_f = 0.5
- TMS din săptămâna 2 până în săptămâna 19 pentru J1: 5, 10, 14, 18, 21, 21.5, 21, 21.5, 17, 14, 11, 11.5, 11.5, 11.5, 11.5, 15, 20, 23
- TMS din săptămâna 2 până în săptămâna 19 pentru J2: 10, 12, 15, 18, 21, 21, 21.2, 21.3, 18, 15, 12, 14, 15, 16, 16, 16, 16, 16
- pentru *p = 3*: cea mai lungă perioadă identificată este s_3-s_{12}; se înregistrează în ordine: *p v-creșteri* consecutive, *p c-stagnări* consecutive, *p v-scăderi* consecutive ale TMS în ambele județe; sunt 3 secvențe de *v-creșteri* / *c-stagnări* / *v-scăderi* în acest interval; J1 înregistrează în această perioadă următoarele valori pentru TMS: 10, 14, 18, 21, 21.5, 21, 21.5, 17, 14, 11, iar J2 - 12, 15, 18, 21, 21, 21.2, 21.3, 18, 15, 12; diferențele între TMS dintr-o săptămână și TMS din săptămâna precedentă, începând cu a doua săptămână din interval, sunt: pentru J1 (4, 4, 3, 0.5, -0.5, 0.5, -4.5, -3, -3), iar pentru J2 (3, 3, 3, 0, 0.2, 0.1, -3.3, -3, -3).

Ale exemple - vezi cod C++.

Este necesară utilizarea subprogramelor care comunică între ele și cu programul principal, precum și specificarea acestora.

Analiză

Identificarea subalgoritmilor



Specificarea subalgoritmilor

Subalgoritmul **citesteParam(p, s, f, vs, vf, cs, cf)**:
Descriere: Citește săptămâna de la care începe analiza, cea în care se încheie analiza, numărul natural *p* și numerele reale *vs, vf, cs, cf* (parametri utilizați în problemă). Perioada analizată trebuie să includă cel puțin 3*p+1 săptămâni consecutive.
Date: -
Rezultate: *p, s, f, vs, vf, cs, cf*
p - număr natural utilizat în problemă, *p* ∈ {1, ..., 17}
s - săptămâna de la care începe analiza, *s* ∈ {1, ..., 52}
f - săptămâna în care se încheie analiza, *f* ∈ {1, ..., 52}
*f - s >= 3*p*
vs - numărul minim de grade cu care poate varia temperatura într-o săptămână față de săptămâna precedentă într-o *v-scădere* sau *v-creștere*, *vs* ∈ (1, 20]
vf - numărul maxim de grade cu care poate varia temperatura într-o săptămână față de săptămâna precedentă într-o *v-scădere* sau *v-creștere*, *vf* ∈ (1, 20]
vs <= vf
cs - numărul minim de grade cu care poate varia temperatura într-o săptămână față de săptămâna precedentă într-o *c-stagnare*, *cs* ∈ [0, 1]

cf - numărul maxim de grade cu care poate varia temperatura într-o săptămână față de săptămâna precedentă într-o *c-stagnare*, *cf* ∈ [0, 1]

cs <= *cf*

Subalgoritmul **citesteSir(T, saptInc, saptSf, n, tipDateT)**:
Descriere: Citește un șir de numere reale, câte un număr pentru fiecare săptămână din perioada determinată de *saptInc* și *saptSf*; calculează lungimea șirului.
Date: *saptInc* - prima săptămână pentru care se citește un număr real
saptSf - ultima săptămână pentru care se citește un număr real
tipDateT - semnificația unui număr real din șir: temperatura medie săptămânală dintr-un anumit județ
Rezultate: *T* - șir de numere reale, câte unul pentru fiecare săptămână aflată în perioada determinată de *saptInc* și *saptSf*; temperatura minimă permisă este -40 °C, iar cea maximă este de 50 °C.
$T = (t_i \in [-40, 50] | i=1..n, n \in N^*)$
n - lungimea șirului de numere citite

Subalgoritmul **detSecvMaxAltern (TJ1, TJ2, lungTemp, p, vs, vf, cs, cf, lungSecvMax, poz)**:
Descriere: Calculează cea mai lungă secvență care începe cu *p v-creșteri* consecutive ale temperaturii medii săptămânale (TMS) și în care:
- *p c-stagnări* consecutive ale TMS sunt precedate de *p v-creșteri* consecutive SAU de *p v-scăderi* consecutive ale TMS;
- *p v-scăderi* consecutive ale TMS sunt precedate în ordine cronologică de *p v-creșteri* consecutive ale TMS și de *p c-stagnări* consecutive ale TMS;
- *p v-creșteri* consecutive ale TMS sunt precedate în ordine cronologică de *p v-scăderi* consecutive ale TMS și de *p c-stagnări* consecutive ale TMS,
în ambele județe analizate.
Date: *TJ1* - un șir de numere reale cu temperatura medie săptămânală în primul județ
TJ2 - un șir de numere reale cu temperatura medie săptămânală în al doilea județ
$TJ1 = (tmsj1_i \in [-40, 50] | i=1..\ lungTemp, lungTemp \in N^*)$
$TJ2 = (tmsj2_i \in [-40, 50] | i=1..\ lungTemp, lungTemp \in N^*)$
lungTemp - lungimea șirurilor cu temperaturile medii săptămânale, *lungTemp* ∈ N*
p - număr natural utilizat în problemă, *p* ∈ {1, ..., 17}
vs, vf, cs, cf - numere reale utilizate în problemă, $v_s, v_f \in (1, 20]$, $v_s <= v_f$; $c_s, c_f \in [0, 1]$
$c_s <= c_f$
Rezultate: *poz* ∈ N* - pentru cea mai lungă secvență care îndeplinește proprietatea cerută, *poz* reprezintă poziția din șir a primei săptămâni din secvență
lungSecvMax ∈ N - lungimea celei mai lungi secvențe care respectă proprietatea precizată; reprezintă numărul de săptămâni din secvență; *lungSecvMax* >= 3*p+1

Subalgoritmul **detSecvMaxAlternPoz (TJ1, TJ2, lungTemp, p, vs, vf, cs, cf, poz, lungSecv)**:
Descriere: Calculează lungimea secvenței care începe cu *p v-creșteri* consecutive ale temperaturii medii săptămânale (TMS) și în care:

- *p c-stagnări* consecutive ale TMS sunt precedate de *p v-creșteri* consecutive SAU de *p v-scăderi* consecutive ale TMS;
- *p v-scăderi* consecutive ale TMS sunt precedate în ordine cronologică de *p v-creșteri* consecutive ale TMS și de *p c-stagnări* consecutive ale TMS;
- *p v-creșteri* consecutive ale TMS sunt precedate în ordine cronologică de *p v-scăderi* consecutive ale TMS și de *p c-stagnări* consecutive ale TMS,

relativ la o poziție dată, în ambele județe analizate.

Date: *TJ1* - un șir de numere reale cu temperatura medie săptămânală în primul județ

TJ2 - un șir de numere reale cu temperatura medie săptămânală în al doilea județ

lungTemp - lungimea șirurilor cu temperaturile medii săptămânale, *lungTemp* $\in$ N*

p - număr natural utilizat în problemă, *p* $\in$ {1, ..., 17}

vs, vf, cs, cf - numere reale utilizate în problemă, v_s, $v_f \in (1, 20]$, $v_s <= v_f$; c_s, $c_f \in [0, 1]$, $c_s <= c_f$

poz $\in$ N - reprezintă poziția din șir începând de la care se verifică proprietatea cerută

Rezultate: *lungSecv* $\in$ N - lungimea secvenței de temperaturii medii săptămânale care îndeplinește proprietatea precizată relativ la poziția dată

Functia **detContInterval(T, vs, vf, cs, cf, semnJ, semnantJ, pozSt, pozSf)**

Descriere: determină dacă șirul T are doar *v-creșteri*, doar *c-stagnări* sau doar *v-scăderi* succesive între pozSt și pozSf+1 și dacă se respectă proprietatea de succesiune în raport cu cele 2 secvențe anterioare de câte p+1 săptămâni consecutive: *p v-creșteri* pot urma doar după *p v-scăderi* succedate de *p c-stagnări*; *p v-scăderi* pot urma doar după *p v-creșteri* succedate de *p c-stagnări*; *p c-stagnări* pot urma doar după *p v-scăderi* sau *p v-creșteri*

Date: *T* - șir de numere reale

vs, vf, cs, cf - numere reale utilizate în problemă, v_s, $v_f \in (1, 20]$, $v_s <= v_f$; c_s, $c_f \in [0, 1]$, $c_s <= c_f$

semnJ - număr întreg care descrie secvența *S*, unde cu *S* se notează secvența anterioară celei analizate

semnantJ - număr întreg care descrie secvența S_{ant}, unde cu S_{ant} se notează secvența anterioară lui *S*

semnJ, semnantJ $\in$ {-1, 0, 1}, cu semnificația:

-1: secvență cu *p v-scăderi* succesive

0: secvență cu *p c-stagnari* succesive

1: secvență cu *p v-creșteri* succesive

pozSt, pozSf $\in$ N*, *pozSf = pozSt + (p-1)*

Rezultate: returnează *true* dacă T are doar *v-creșteri*, doar *c-stagnări* sau doar *v-scăderi* succesive între pozSt și pozSf+1 și dacă se respectă proprietatea de succesiune, *false* altfel

semnJ - număr întreg care descrie tipul de secvență dintre pozițiile pozSt și pozSf+1

semnantJ - număr întreg care descrie secvența *S*, unde cu *S* se notează secvența anterioară celei analizate

semnJ, semnantJ $\in$ {-1, 0, 1}, cu semnificația:

-1: secvență cu *p v-scăderi* succesive

0: secvență cu *p c-stagnari* succesive

1: secvență cu *p v-creșteri* succesive

Subalgoritmul **afiseazaSecvMax(TJ1, TJ2, lungTemp, inceputInterval, lungimeSecventa, saptInceput, p)**:
Descriere: Afișează perioada de timp solicitată în problemă.
Date: *TJ1* - un șir de numere reale cu temperatura medie săptămânală în primul județ
TJ2 - un șir de numere reale cu temperatura medie săptămânală în al doilea județ
TJ1 = ($tmsj1_i \in [-40, 50] | i=1..$ *lungTemp*, *lungTemp* $\in$ N*)
TJ2 = ($tmsj2_i \in [-40, 50] | i=1..$ *lungTemp*, *lungTemp* $\in$ N*)
lungTemp - lungimea șirurilor cu temperaturile medii săptămânale, *lungTemp* $\in$ N*
inceputInterval $\in$ N* - poziția de la care începe secvența identificată; *inceputInterval* $\in$ {1, ..., *lungTemp*}
lungimeSecventa $\in$ N - lungimea secvenței care trebuie afișată
saptInceput $\in$ N - săptămâna de început a perioadei analizate
p - număr natural utilizat în problemă, *p* $\in$ {1, ..., 17}
Rezultate: se afișează intervalul solicitat în problemă

Proiectare

```
subalgoritmul citesteParam(p, s, f, vs, vf, cs, cf) este:
	repeta
		@tipareste „ Introduceti valoarea parametrului p: ”
		@citeste p
	pana-cand p >= 1 AND p <= 17

	repeta
		@tipareste „ Introduceti saptamana de start s si saptamana de final f a perioadei analizate (f - s >= 3 * p): ”
		repeta
			@tipareste „ Introduceti s: ”
			@citeste s
		pana-cand s >= 1 AND s <= 52
		repeta
			@tipareste „ Introduceti f: ”
			@citeste f
		pana-cand f >= 1 AND f <= 52
	pana-cand f - s >= 3 * p

	@tipareste „ Introduceti valoarea parametrilor vs si vf (vs, vf in (1, 20]):”
	repeta
		@tipareste „ Introduceti valoarea parametrului vs: ”
		@citeste vs
	pana-cand vs > 1 AND vs <= 20
	repeta
		@tipareste „ Introduceti valoarea parametrului vf: ”
		@citeste vf
	pana-cand vf > 1 AND vf <= 20 AND vf >= vs

	@tipareste „ Introduceti valoarea parametrilor cs si cf (cs, cf in [0, 1]):”
	repeta
		@tipareste „ Introduceti valoarea parametrului cs: ”
		@citeste cs
	pana-cand cs >= 0 AND cs <= 1
```

```
	repeta
		@tipareste „ Introduceti valoarea parametrului cf: ”
		@citeste cf
	pana-cand cf >= 0 AND cf <= 1 AND cf >= cs
sf-subalgoritm

subalgoritmul citesteSir(T, saptInc, saptSf, n, tipDateT) este:
	n ← saptSf – saptInc + 1
	pentru i ← 1, n executa
		repeta
			@tipareste tipDateT, " in saptamana ”, saptInc+i-1, ": "
			@citeste T[i]
		pana-cand T[i] >= -40 AND T[i] <= 50
	sf-pentru
sf-subalgoritm

subalgoritmul detSecvMaxAltern(TJ1, TJ2, lungTemp, p, vs, vf, cs, cf, lungSecvMax, poz)
este:
	i ← 1
	cat-timp i <= lungTemp - p executa
		lungSecv ← 0
		detSecvMaxAlternPoz(TJ1, TJ2, lungTemp, p, vs, vf, cs, cf, i, lungSecv)
		daca lungSecv > lungSecvMax atunci
			lungSecvMax ← lungSecv
			poz ← i
		sf-daca
		daca lungSecv = 0 atunci
			i ← i + 1
		altfel
			i ← i + lungSecv - (p+1)
		sf-daca
	sf-cat-timp
sf-subalgoritm

subalgoritmul detSecvMaxAlternPoz(TJ1, TJ2, lungTemp, p, vs, vf, cs, cf, poz, lungSecv)
este:
	 lungSecv ← 0
	 i ← poz
	 contor ← 0
	 cont ← true
	semnJ1 ← 0, semnJ2 ← 0
	semnantJ1 ← -1, semnantJ2 ← -1
	cat-timp i <= lungTemp - p AND cont executa
		daca detContInterval(TJ1, vs, vf, cs, cf, semnJ1, semnantJ1, i,
			i + (p-1)) AND
		     detContInterval(TJ2, vs, vf, cs, cf, semnJ2, semnantJ2, i,
			i + (p-1)) atunci
			  i ← i + p
			  contor ← contor + 1
		altfel
			cont ← false
		sf-daca
	sf-cat-timp
	daca contor >= 3 atunci
		lungSecv ← i - poz + 1
	sf-daca
sf-subalgoritm

functia detContInterval(T, vs, vf, cs, cf, semnJ, semnantJ, pozSt, pozSf) este:
```

```
	i ← pozSt
	cont ← true

	daca semnJ = -1 OR semnJ = 1 atunci
		cat-timp i <= pozSf AND cont executa
			daca |T[i+1]-T[i]| < cs OR |T[i+1]-T[i]| > cf atunci
				cont ← false
			altfel
				i ← i+1
			sf-daca
		sf-cat-timp
		daca cont = true atunci
			daca semnJ = -1 atunci
				semnantJ = -1
			altfel
				semnantJ = 1
			sf-daca
			semnJ = 0
		sf-daca
	altfel
		daca semnJ = 0 atunci
			daca semnantJ = -1 atunci
				cat-timp i <= pozSf AND cont executa
					daca T[i+1]-T[i] < vs OR T[i+1]-T[i] > vf atunci
						cont ← false
					altfel
						i ← i+1
					sf-daca
				sf-cat-timp
				daca cont = true atunci
					semnantJ = 0
					semnJ = 1
				sf-daca
			altfel
				daca semnantJ = 1 atunci
					cat-timp i <= pozSf AND cont executa
						daca T[i]-T[i+1] < vs OR T[i]-T[i+1] > vf
						    atunci
							cont ← false
						altfel
							i ← i+1
						sf-daca
					sf-cat-timp
					daca cont = true atunci
						semnantJ = 0
						semnJ = -1
					sf-daca
				sf-daca
			sf-daca
		sf-daca
	sf-daca
	detContInterval ← cont
sf-functie


subalgoritmul afiseazaSecvMax(TJ1, TJ2, lungTemp, inceputInterval, lungimeSecventa,
saptInceput, p) este:
	daca inceputInterval = 0 atunci
		@tipareste "Nu exista niciun interval de timp cu proprietatea dorita. "
```

```
	altfel
		saptIncInterval ← saptInceput + inceputInterval - 1
		saptSfInterval ← saptIncInterval + lungimeSecventa - 1
		@tipareste "Cea mai lunga perioada care indeplineste proprietatea ceruta
			este: ", saptIncInterval , " - ", saptSfInterval
		@tipareste "Nr de v-cresteri / c-stagnari / v-scaderi in perioada respectiva
	este: ", (saptSfInterval - saptIncInterval)/p

		@tipareste "Diferente intre temperaturile medii saptamanale din 2 saptamani
	consecutive in perioada considerata, pentru ambele judete: "
		i ← inceputInterval + 1
		cat-timp i <= inceputInterval + lungSecv - 1 executa
			@tipareste TJ1[i] - TJ1[i-1], "  "
			@tipareste TJ2[i] - TJ2[i-1], "  "
			i ← i + 1
		sf-cat-timp
	sf-daca
sf-subalgoritm

Algoritmul MonitorizareClima este:
	s ← 0
	f ← 0
	p ← 0
	vs ← 0
	vf ← 0
	cs ← 0
	cf ← 0
	lungTemp ← 0

	citesteParam(p, s, f, vs, vf, cs, cf)
	citesteSir(TJ1, s, f, lungTemp, " temperatura medie saptamanala in judetul J1")
	citesteSir(TJ2, s, f, lungTemp, " temperatura medie saptamanala in judetul J2")

	lungSecv ← 0
	poz ← 0
	detSecvMaxAltern(TJ1, TJ2, lungTemp, p, vs, vf, cs, cf, lungSecv, poz)
	afiseazaSecvMax(TJ1, TJ2, lungTemp, poz, lungSecv, s, p)
Sf-Algoritm
```

Exemple

Date de intrare	Rezultate
vezi enunț și cod sursă C++ / Pascal	

Cod sursă C++

```
#include "pch.h"
#include <iostream>
using namespace std;


void citesteParam(int& p, int& s, int& f, float& vs, float& vf, float& cs, float& cf)
```

```
{
	do
	{
		cout << "Introduceti valoarea parametrului p: ";
		cin >> p;
	} while (p < 1 || p > 17);

	do
	{
		cout << "Introduceti saptamana de start s si saptamana de final f a perioadei
analizate (f - s >= 3 * p): ";
		do
		{
			cout << "Introduceti s: ";
			cin >> s;
		} while (s < 1 || s > 52);
		do
		{
			cout << "Introduceti f: ";
			cin >> f;
		} while (f < 1 || f > 52);
	} while (f - s < 3 * p);

	cout << "Introduceti valoarea parametrilor vs si vf (vs, vf in (1, 20]): ";
	do
	{
		cout << "Introduceti vs: ";
		cin >> vs;
	} while (vs <= 1 || vs > 20);
	do
	{
		cout << "Introduceti vf: ";
		cin >> vf;
	} while (vf <= 1 || vf > 20 || vf < vs);

	cout << "Introduceti valoarea parametrilor cs si cf (cs, cf in [0, 1]): ";
	do
	{
		cout << "Introduceti cs: ";
		cin >> cs;
	} while (cs < 0 || cs > 1);
	do
	{
		cout << "Introduceti cf: ";
		cin >> cf;
	} while (cf < 0 || cf > 1 || cf < cs);
}

void citesteSir(float T[], int saptInc, int saptSf, int& n, const char tipDateT[50])
{
	n = saptSf - saptInc + 1; // se citeste un sir de numere reale
	for (int i = 0; i < n; i++)
	{
		do
```

```
		{
			cout << tipDateT << " in saptamana " << saptInc + i << ": ";
			cin >> T[i];
		} while (T[i] < -40 || T[i] > 50);
	}
}

bool detContInterval(float T[], float vs, float vf, float cs, float cf, int& semnJ, int&
semnantJ, int pozSt, int pozSf)
{
	int i = pozSt;
	bool cont = true;

	if (semnJ == -1 || semnJ == 1) //daca anterior in J s-a inregistrat o perioada cu
					//v-scaderi sau v-cresteri, acum trebuie sa urmeze o
					//perioada cu c-stagnari
	{
		while (i <= pozSf && cont)
			if (abs(T[i + 1] - T[i]) < cs || abs(T[i + 1] - T[i]) > cf)
				cont = false;
			else i++;
		if (cont)
		{
			if (semnJ == -1) semnantJ = -1;
				else semnantJ = 1;   //semnJ == 1
			semnJ = 0;                  //s-a identificat o perioada cu c-stagnari
		}
	}
	else if (semnJ == 0) //daca anterior in J s-a inregistrat o perioada cu c-stagnari
		if (semnantJ == -1) //daca anterior perioadei anterioare de c-stagnari din J s-a
		inregistrat o perioada cu v-scaderi, trebuie sa urmeze o perioada cu v-cresteri
		{
			while (i <= pozSf && cont)
				if (T[i + 1] - T[i] < vs || T[i + 1] - T[i] > vf)
					cont = false;
				else i++;
			if (cont)
			{
				semnantJ = 0;
				semnJ = 1;//s-a identificat o perioada cu v-cresteri
			}
		}
		else if (semnantJ = 1) //daca anterior perioadei anterioare de c-stagnari din J
s-a inregistrat o perioada cu v-cresteri, trebuie sa urmeze o perioada cu v-scaderi
		{
			while (i <= pozSf && cont)
				if (T[i] - T[i + 1] < vs || T[i] - T[i + 1] > vf)
					cont = false;
				else i++;
			if (cont)
			{
				semnantJ = 0;
				semnJ = -1;//s-a identificat o perioada cu v-scaderi
			}
```

```
			}
		return cont;
}

void detSecvMaxAlternPoz(float TJ1[], float TJ2[], int lungTemp, int p, float vs, float vf,
float cs, float cf, int poz, int& lungSecv)
{
		lungSecv = 0;
		int i = poz;
		int contor = 0;
		bool cont = true;
		int semnJ1 = 0, semnJ2 = 0; //prima secventa trebuie sa contina v-cresteri; deci se
presupune existenta unei perioade imediat anterioare cu c-stagnari,
		int semnantJ1 = -1, semnantJ2 = -1; //iar anterior acesteia – a unei perioade cu v-
scaderi
		while (i < lungTemp - p && cont)
		{
			if (detContInterval(TJ1, vs, vf, cs, cf, semnJ1, semnantJ1, i, i + (p - 1)) &&
detContInterval(TJ2, vs, vf, cs, cf, semnJ2, semnantJ2, i, i + (p - 1)))
			{
				i += p;
				contor++;
			}
			else
				cont = false;
		}
		if (contor >= 3)
			lungSecv = i - poz + 1;
}

void detSecvMaxAltern(float TJ1[], float TJ2[], int lungTemp, int p, float vs, float vf,
float cs, float cf, int& lungSecvMax, int& poz)
{
		int i = 0;
		while (i < (lungTemp - p))
		{
			int lungSecv = 0;
			detSecvMaxAlternPoz(TJ1, TJ2, lungTemp, p, vs, vf, cs, cf, i, lungSecv);
			if (lungSecv > lungSecvMax)
			{
				lungSecvMax = lungSecv;
				poz = i;
			}
			if (lungSecv == 0)
				i++;
			else
				i += lungSecv - (p + 1);
		}
}

void afiseazaSecvMax(float TJ1[], float TJ2[], int lungTemp, int inceputInterval, int
lungimeSecventa, int saptInceput, int p)
{
		if (inceputInterval == -1)
```

```
	{
		cout << "Nu exista niciun interval de timp cu proprietatea dorita." << endl;
		return;
	}
	int saptIncInterval = saptInceput + inceputInterval;
	int saptSfInterval = saptIncInterval + lungimeSecventa - 1;

	cout << "Cea mai lunga perioada care indeplineste proprietatea ceruta este: " <<
saptIncInterval << " - " << saptSfInterval << endl;
	cout << "Nr de v-cresteri / c-stagnari / v-scaderi  in perioada respectiva este: " <<
(saptSfInterval - saptIncInterval) / p << endl;

	int i;
	cout << "Diferente intre temperaturile medii saptamanale din 2 saptamani consecutive in
perioada considerata, pentru ambele judete: " << endl;
	i = inceputInterval + 1;
	while (i <= inceputInterval + lungimeSecventa - 1)
	{
		cout << "temp[" << i << "] - temp[" << i - 1 << "] ";
		cout << "(J1):   ";
		cout << TJ1[i] - TJ1[i - 1] << "  " << endl;

		i++;
	}

	i = inceputInterval + 1;
	while (i <= inceputInterval + lungimeSecventa - 1)
	{
		cout << "temp[" << i << "] - temp[" << i - 1 << "] ";
		cout << "(J2):   ";
		cout << TJ2[i] - TJ2[i - 1] << "  " << endl;

		i++;
	}
}


int main()
{
	float TJ1[52], TJ2[52];
	int p, s, f, lungTemp, lungSecv, poz;
	float vs, vf, cs, cf;
	char tipDate[20];

	s = 0, f = 0, p = 0, vs = 0, vf = 0, cs = 0, cf = 0, lungTemp = 0; //citire param, temp
	citesteParam(p, s, f, vs, vf, cs, cf);
	citesteSir(TJ1, s, f, lungTemp, "temperatura medie saptamanala in judetul J1");
	citesteSir(TJ2, s, f, lungTemp, "temperatura medie saptamanala in judetul J2");
	//e1
	/*s = 2, f = 19, vs = 3, vf = 7, cs = 0, cf = 0.5, p = 3; lungTemp = 18;
	float TJ1[18] = { 5, 10, 14, 18, 21, 21.5, 21, 21.5, 17, 14, 11, 11.5, 11.5, 11.5,
11.5, 15, 20, 23};
	float TJ2[18] = { 10, 12, 15, 18, 21, 21, 21.2, 21.3, 18, 15, 12, 14, 15, 16, 16, 16,
16, 16};*/
```

```
        //e2
        /*s = 1, f = 8, vs = 3, vf = 7, cs = 0, cf = 0.5, p = 2; lungTemp = 8;
        float TJ1[18] = { 0, 4, 8, 12, 12, 12, 8, 4 };
        float TJ2[18] = { 0, 4, 8, 12, 12, 12, 8, 4 };*/

        //e3
        /*s = 1, f = 7, vs = 3, vf = 7, cs = 0, cf = 0.5, p = 2; lungTemp = 7;
        float TJ1[18] = { 20, 20, 20, 16, 12, 12, 12 };
        float TJ2[18] = { 20, 20, 20, 16, 12, 12, 12 };*/

        //e4
        /*s = 1, f = 9, vs = 3, vf = 7, cs = 0, cf = 0.5, p = 2; lungTemp = 9;
        float TJ1[18] = { 10, 10, 10, 14, 18, 18, 18, 14, 10 };
        float TJ2[18] = { 10, 10, 10, 14, 18, 18, 18, 14, 10 };*/

        //e5
        /*s = 1, f = 11, vs = 3, vf = 7, cs = 0, cf = 0.5, p = 2; lungTemp = 11;
        float TJ1[18] = { 20, 16, 12, 12, 12, 16, 20, 20, 20, 16, 12 };
        float TJ2[18] = { 20, 16, 12, 12, 12, 16, 20, 20, 20, 16, 12 };*/

        //e6
        /*s = 1, f = 7, vs = 3, vf = 7, cs = 0, cf = 0.5, p = 2; lungTemp = 7;
        float TJ1[18] = { 10, 14, 18, 18, 18, 22, 26 };
        float TJ2[18] = { 10, 14, 18, 18, 18, 22, 26 };*/

        //e7
        /*s = 1, f = 7, vs = 3, vf = 7, cs = 0, cf = 0.5, p = 2; lungTemp = 7;
        float TJ1[18] = { 10, 14, 18, 18, 18, 14, 10 };
        float TJ2[18] = { 10, 14, 18, 18, 18, 14, 10 };*/

        //e8
        /*s = 1, f = 16, vs = 3, vf = 7, cs = 0, cf = 0.5, p = 2; lungTemp = 16;
        float TJ1[18] = { 10, 14, 18, 18, 18, 18, 14, 10, 10, 10, 14, 18, 18, 18, 14, 10 };
        float TJ2[18] = { 10, 14, 18, 18, 18, 18, 14, 10, 10, 10, 14, 18, 18, 18, 14, 10 };*/

        //e9
        /*s = 1, f = 12, vs = 3, vf = 7, cs = 0, cf = 0.5, p = 2; lungTemp = 12;
        float TJ1[18] = { 10, 14, 18, 18, 18, 18, 22, 26, 26, 26, 22, 18 };
        float TJ2[18] = { 10, 14, 18, 18, 18, 18, 22, 26, 26, 26, 22, 18 };*/

        //e10
        /*s = 1, f = 32, vs = 3, vf = 7, cs = 0, cf = 0.5, p = 2; lungTemp = 32;
        float TJ1[32] = { 10, 14, 18, 18, 18, 14, 10, 10, 10, 14, 18, 22, 22, 22, 18, 14, 14,
14, 18, 22, 22, 22, 18, 14, 14, 14, 18, 22, 22, 22, 18, 14 };
        float TJ2[32] = { 10, 14, 18, 18, 18, 14, 10, 10, 10, 14, 18, 22, 22, 22, 18, 14, 14,
14, 18, 22, 22, 22, 18, 14, 14, 14, 18, 22, 22, 22, 18, 14 };*/

        // identificam cea mai lunga secventa cu proprietatea ceruta
        lungSecv = 0;
        poz = -1;
        detSecvMaxAltern(TJ1, TJ2, lungTemp, p, vs, vf, cs, cf, lungSecv, poz);

        afiseazaSecvMax(TJ1, TJ2, lungTemp, poz, lungSecv, s, p);
```

```
}
```

Cod sursă Pascal

```
Program Clima;
type
tablou = array[1..52] of real;

procedure citesteParam(var p, s, f:integer; var vs, vf, cs, cf: real );
begin
repeat
    writeln('Introduceti valoarea parametrului p: ');
    read(p);
until (p >= 1) and (p <= 17);
repeat
    writeln('Intoduceti saptamana de start s si saptamana de final f a perioadei
analizate (f-s >= 3*p): ');
    repeat
        writeln('Introduceti s: ');
        read(s);
    until (s >= 1) and (s <= 52);
    repeat
        writeln('Introduceti f: ');
        read(f);
    until (f >= 1) and (f <= 52);
until (f - s >= 3 * p);
repeat
    writeln('Introduceti valoarea parametrilor vs si vf (vs, vf in (1, 20]): ');
    repeat
        writeln('Introduceti vs: ');
        read(vs);
    until (vs > 1) and (vs <= 20);
    repeat
        writeln('Introduceti vf: ');
        read(vf);
    until (vf > 1) and (vf <= 20);
until (vs <= vf);
repeat
    writeln('Introduceti valoarea parametrilor cs si cf (cs, cf in [0, 1]): ');
    repeat
        writeln('Introduceti cs: ');
        read(cs);
    until (cs >= 0) and (cs <= 1);
    repeat
        writeln('Introduceti cf: ');
        read(cf);
    until (cf >= 0) and (cf <= 1);
until (cs <= cf);
end;
```

```
procedure citesteSir(var T: tablou; saptInc, saptSf: integer; var n: integer;
tipDateT: string);
var i: integer;
nr: string;
begin
    n := saptSf - saptInc + 1 ;
    for i := 1 to n do
    begin
        repeat
            str((i + saptInc - 1), nr);
            writeln(tipDateT + ' in saptamana ' + nr + ': ');
            read(T[i]);
        until (T[i] >= -40) and (T[i] <= 50);
    end;
end;

function detContInterval(T: tablou; vs, vf, cs, cf: real; var semnJ, semnantJ:
integer; pozSt, pozSf: integer): Boolean;
var i: integer;
dif: real;
cont: Boolean;
begin
    i := pozSt;
    cont := True;

    if (semnJ = -1) or (semnJ = 1) then
    //daca anterior s-a inregistrat o perioada cu v-scaderi sau v-cresteri
    //acum ne trebuie o perioada cu c-stagnari
    begin
        while (i <= pozSf) and cont do
        begin
            dif := abs(T[i+1] - T[i]);
            if (dif < cs) or (dif > cf) then
                cont := False
            else
                i := i + 1;
        end;
        //daca perioada contine doar c-stagnari, schimbam semnele pentru
perioadele anterioare
        if cont then
        begin
            semnantJ := semnJ;
            semnJ := 0;
        end;
    end
    else if (semnJ = 0) then //perioada anterioara a continut doar c-stagnari
    begin
        if (semnantJ = -1) then //inainte de perioada cu c-stagnari a existat o
perioada cu v-scaderi, acum ar trebui sa urmeze o perioada cu v-cresteri
        begin
            while (i <= pozSf) and cont do
            begin
                dif := T[i+1] - T[i];
```

```
                if (dif < vs) or (dif > vf) then
                    cont := False
                else
                    i := i + 1;
            end;
            //daca exista doar v-cresteri, schimbam semnele pentru perioadele
anterioare
            if cont then
            begin
                semnantJ := semnJ;
                semnJ := 1; // crestere
            end;
        end
        else if (semnantJ = 1) then // perioada anterioara a continut v-cresteri,
acum ar trebui sa urmeze v-scaderi
        begin
            while (i <= pozSf) and cont do
            begin
                dif  := T[i] - T[i + 1];
                if (dif < vs) or (dif > vf) then
                    cont := False
                else
                    i := i + 1;
            end;
            if cont then
            begin
                semnantJ := semnJ;
                semnJ := -1; //scadere
            end;
        end;
    end;
    detContInterval := cont;
end;

procedure detSecvMaxAlternPoz(TJ1, TJ2: tablou; lungTemp, p: integer; vs, vf, cs,
cf: real; poz: integer; var lungSecv: integer);
var
    i, contor, semnJ1, semnJ2, semnAntJ1, semnAntJ2: integer;
    cont: Boolean;
begin
    lungSecv := 0;
    i := poz;
    semnJ1 := 0; //prima secventa trebuie sa contina v-cresteri.
    semnJ2 := 0;  // setam o perioada fictive cu c-stagnari si...
    semnAntJ1 := -1;  // o perioada fictive cu v-scaderi
    semnAntJ2 := -1; // pentru ambele judete
    cont := True;
    contor := 0;
    //vom verifica o secventa care incepe de la i. Trebuie sa avem minim inca p
pozitii pana la finalul tablourilor
    while (i <= lungTemp - p) and cont do
    begin
```

```
        if (detContInterval (TJ1, vs, vf, cs, cf, semnJ1, semnAntJ1, i, i+p- 1))
and (detContInterval(TJ2, vs, vf, cs, cf, semnJ2, semnAntJ2, i, i+p-1)) then
        begin
            i := i + p;
            contor := contor + 1;
        end else
            cont := False;
    end;
    if (contor >= 3) then
        lungSecv := i - poz + 1;

end;


procedure detSecvMaxAltern(TJ1, TJ2: tablou; lungTemp, p: integer; vs, vf, cs,
cf: real; var lungSecvMax, poz: integer);
var
    i, lungSecv: integer;
begin
    i := 1;
    while ( i <= (lungTemp - p)) do
    begin
        lungSecv := 0;
        detSecvMaxAlternPoz(TJ1, TJ2, lungTemp, p, vs, vf, cs, cf, i, lungSecv);
        if (lungSecv > lungSecvMax) then
        begin
            lungSecvMax := lungSecv;
            poz := i;
        end;
        if (lungSecv = 0) then
            i := i + 1
        else
            i := i + (lungSecv - (p + 1));
    end;
end;

procedure afiseazaSecvMax(TJ1, TJ2: tablou; lungTemp, inceputInterval,
lungimeSecventa, saptInceput, p: integer);
var
    i, saptIncInterval, saptSfInterval: integer;
    s1, s2, s3: string; //pentru transformarea numerelor in stringuri
begin
    if (inceputInterval = 0) then
        writeln('Nu exista niciun interval de timp cu proprietatea dorita')
    else
    begin
        saptIncInterval := saptInceput + inceputInterval-1;
        saptSfInterval := saptIncInterval + lungimeSecventa-1;
        str(saptIncInterval, s1);
        str(saptSfInterval, s2);
        writeln('Cea mai lunga perioada care indeplineste proprietatea ceruta
este : ' + s1 + ' - ' + s2);
        str((saptSfInterval -  saptIncInterval)/ p, s3);
```

```
        writeln('Nr de v-crester / c-stagnari / v-scaderi in perioada respectiva
este: ' + s3);
        writeln('Diferenta intre temperaturile medii saptamanale din 2 saptamani
consecutive in perioada considerata, pentru ambele judete: ');
       i := inceputInterval + 1;
       while (i <= inceputInterval + lungimeSecventa - 1) do
       begin
            str(i, s1);
            str((i-1), s2);
            str((TJ1[i] - TJ1[i-1]), s3);
            writeln('temp['+ s1 + '] - temp[' + s2 + '] (J1)  : ' + s3);
            i := i + 1;
       end;
        i := inceputInterval + 1;
        while (i <= inceputInterval + lungimeSecventa - 1) do
        begin
            str(i, s1);
            str((i-1), s2);
            str((TJ2[i] - TJ2[i-1]), s3);
            writeln('temp['+ s1 + '] - temp[' + s2 + '] (J2)  : ' + s3);
            i := i + 1;
       end;
    end;
end;
//PROGRAM PRINCIPAL
var
//decomentati partea pana la Exemplu 1 pentru a citi de la tastatura / comentati
aceasta parte pentru a folosi exemple initializate
p, s, f, n, lungTemp, lungSecv, poz: integer;
vs, vf, cs, cf: real;
TJ1, TJ2: tablou;

//Exemplu 1 - date de intrare initializate:
{lungSecv, poz: integer;
s : integer = 2;
f : integer = 19;
p : integer = 3;
vs : real = 3;
vf : real = 7;
cs : real = 0;
cf : real = 0.5;
lungTemp : integer = 18;
TJ1: array[1..52] of real = (5, 10, 14, 18, 21, 21.5, 21, 21.5, 17, 14, 11, 11.5,
11.5, 11.5, 11.5, 15, 20, 23, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
TJ2: array[1..52] of real = (10, 12, 15, 18, 21, 21, 21.2, 21.3, 18, 15, 12, 14,
15, 16, 16, 16, 16, 16, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
}

//Exemplu 2 - date de intrare initializate
{lungSecv, poz: integer;
s : integer = 1;
```

```
f : integer = 8;
p : integer = 2;
vs : real = 3;
vf : real = 7;
cs : real = 0;
cf : real = 0.5;
lungTemp : integer = 8;
TJ1: array[1..52] of real = (0, 4, 8, 12, 12, 12, 8, 4, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0);
TJ2: array[1..52] of real = (0, 4, 8, 12, 12, 12, 8, 4, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0);
}

//Exemplu 3 - date de intrare initializate
{lungSecv, poz: integer;
s : integer = 1;
f : integer = 7;
p : integer = 2;
vs : real = 3;
vf : real = 7;
cs : real = 0;
cf : real = 0.5;
lungTemp : integer = 7;
TJ1: array[1..52] of real = (20, 20, 20, 16, 12, 12, 12, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
TJ2: array[1..52] of real = (20, 20, 20, 16, 12, 12, 12, 0, 0, 0, 0, 0, 0, 0, 0,
0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
}

//Exemplu 4 - date de intrare initializate
{lungSecv, poz: integer;
s : integer = 1;
f : integer = 9;
p : integer = 2;
vs : real = 3;
vf : real = 7;
cs : real = 0;
cf : real = 0.5;
lungTemp : integer = 9;
TJ1: array[1..52] of real = (10, 10, 10, 14, 18, 18, 18, 14, 10, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
TJ2: array[1..52] of real = (10, 10, 10, 14, 18, 18, 18, 14, 10, 0, 0, 0, 0, 0,
0, 0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
}

//Exemplu 5 - date de intrare initializate
{lungSecv, poz: integer;
```

```
s : integer = 1;
f : integer = 11;
p : integer = 2;
vs : real = 3;
vf : real = 7;
cs : real = 0;
cf : real = 0.5;
lungTemp : integer = 11;
TJ1: array[1..52] of real = (20, 16, 12, 12, 12, 16, 20, 20, 20, 16, 12, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
TJ2: array[1..52] of real = (20, 16, 12, 12, 12, 16, 20, 20, 20, 16, 12, 0, 0, 0,
0, 0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
}

//Exemplu 6 - date de intrare initializate
{lungSecv, poz: integer;
s : integer = 1;
f : integer = 7;
p : integer = 2;
vs : real = 3;
vf : real = 7;
cs : real = 0;
cf : real = 0.5;
lungTemp : integer = 7;
TJ1: array[1..52] of real = (10, 14, 18, 18, 18, 22, 26, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
TJ2: array[1..52] of real = (10, 14, 18, 18, 18, 22, 26, 0, 0, 0, 0, 0, 0, 0, 0,
0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
}

//Exemplu 7 - date de intrare initializate
{lungSecv, poz: integer;
s : integer = 1;
f : integer = 7;
p : integer = 2;
vs : real = 3;
vf : real = 7;
cs : real = 0;
cf : real = 0.5;
lungTemp : integer = 7;
TJ1: array[1..52] of real = (10, 14, 18, 18, 18, 14, 10, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
TJ2: array[1..52] of real = (10, 14, 18, 18, 18, 14, 10, 0, 0, 0, 0, 0, 0, 0, 0,
0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
}

//Exemplu 8 - date de intrare initializate
```

```
{lungSecv, poz: integer;
s : integer = 1;
f : integer = 16;
p : integer = 2;
vs : real = 3;
vf : real = 7;
cs : real = 0;
cf : real = 0.5;
lungTemp : integer = 16;
TJ1: array[1..52] of real = (10, 14, 18, 18, 18, 18, 14, 10, 10, 10, 14, 18, 18,
18, 14, 10, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
TJ2: array[1..52] of real = (10, 14, 18, 18, 18, 18, 14, 10, 10, 10, 14, 18, 18,
18, 14, 10,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
}

//Exemplu 9 - date de intrare initializate
{lungSecv, poz: integer;
s : integer = 1;
f : integer = 12;
p : integer = 2;
vs : real = 3;
vf : real = 7;
cs : real = 0;
cf : real = 0.5;
lungTemp : integer = 12;
TJ1: array[1..52] of real = (10, 14, 18, 18, 18, 18, 22, 26, 26, 26, 22, 18, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
TJ2: array[1..52] of real = (10, 14, 18, 18, 18, 18, 22, 26, 26, 26, 22, 18, 0,
0, 0, 0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
}

//Exemplu 10 - date de intrare initializate
{lungSecv, poz: integer;
s : integer = 1;
f : integer = 32;
p : integer = 2;
vs : real = 3;
vf : real = 7;
cs : real = 0;
cf : real = 0.5;
lungTemp : integer = 32;
TJ1: array[1..52] of real = (10, 14, 18, 18, 18, 14, 10, 10, 10, 14, 18, 22, 22,
22, 18, 14, 14, 14, 18, 22, 22, 22, 18, 14, 14, 14, 18, 22, 22, 22, 18, 14, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
TJ2: array[1..52] of real = (10, 14, 18, 18, 18, 14, 10, 10, 10, 14, 18, 22, 22,
22, 18, 14, 14, 14, 18, 22, 22, 22, 18, 14, 14, 14, 18, 22, 22, 22, 18, 14, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
}
begin
```

```
    //citire date - decomentati aceasta parte pentru a citi de la tastatura /
comentati aceasta parte daca folositi exemple initializate in cod
    citesteParam(p, s, f, vs, vf, cs, cf);
    citesteSir(TJ1, s, f, lungTemp, 'Temperatura medie saptamanala in judetul
J1');
    citesteSir(TJ2, s, f, lungTemp, 'Temperatura medie saptamanala in judetul
J2');
    //final citire date

    lungSecv := 0;
    poz := 0;
    detSecvMaxAltern(TJ1, TJ2, lungTemp, p, vs, vf, cs, cf, lungSecv, poz);
    afiseazaSecvMax(TJ1, TJ2, lungTemp, poz, lungSecv, s, p);
end.
```

Problema 3 - Molecula

Enunț

O moleculă este alcătuită dintr-o secvență de atomi. Descrierea sa se face cu ajutorul unei formule alcătuite din litere care reprezintă atomii din moleculă. Exemplu: H reprezintă hidrogen, C reprezintă carbon, iar O - oxigen; COOH este o moleculă formată dintr-un atom de carbon, 2 atomi de oxigen și un atom de hidrogen.
Descrierea compactă a formulelor presupune că:

- unii atomi se pot grupa prin adăugarea parantezelor; de exemplu, CH(OH) conține grupa (OH);
- o grupă poate conține alte grupe;
- 2 sau mai multe apariții consecutive ale unei litere pot fi înlocuite de litera respectivă urmată de numărul de apariții ale acesteia; de exemplu, COOH se poate rescrie ca CO2H;
- 2 sau mai multe apariții consecutive ale unei grupe pot fi înlocuite de grupa respectivă urmată de numărul său de apariții; de exemplu CH(CO2H)3 conține în total 4 atomi de carbon, 6 atomi de oxigen și 4 atomi de hidrogen;
- numărul care reprezintă frecvența (unui atom, unei grupe de atomi) este mai mare decât 1 și mai mic decât 10.

Masa unei molecule este suma maselor atomilor care alcătuiesc molecula. Masa unui atom de hidrogen este 1, cea a unui atom de carbon este 12, iar masa unui atom de oxigen este 16.
Calculați masa unei molecule descrisă printr-o formulă de cel mult 100 de caractere.

Analiză

Problema este similară cu cea a evaluării unei expresii aritmetice. Se utilizează un tablou care simulează o stivă.
Se parcurge formula. Dacă se întâlnește:

- o literă (C, O, H), se adaugă la capătul tabloului masa atomului corespunzător;
- o paranteză deschisă, se adaugă la capătul tabloului valoarea -1 (simbol special pentru reprezentarea unei paranteze deschise);
- o paranteză închisă, se șterg și se însumează elemente începând de la capătul tabloului și avansând spre începutul acestuia până la întâlnirea valorii -1 (paranteza deschisă); se introduce suma obținută la capătul tabloului;
- o cifră, se șterge ultimul element din tablou, se înmulțește elementul eliminat cu cifra din formulă și se adaugă rezultatul la capătul tabloului;

Când se finalizează parcurgerea formulei, se însumează toate elementele din tablou.

Obs: stiva necesară în problemă se poate simula ușor cu ajutorul unui tablou și a lungimii acestuia.

Specificarea subalgoritmului

Funcție **masaMolecula(formula):**

Descriere: calculează masa unei molecule descrise de *formula*
Date: formula - șir de caractere care descrie o formulă conform regulilor din cerința problemei
Rezultate: masa formulei

Proiectare

```
funcție masaMolecula(formula) este:
	lungime ← 1
	pentru i ← 1, dimensiune(formula) executa
		c ← formula[i]
		daca c = 'O' atunci
			stiva[lungime] ← 16
			lungime ← lungime + 1
		altfel daca c = 'C' atunci
				stiva[lungime] ← 12
				lungime ← lungime + 1
			altfel daca c = 'H' atunci
					stiva[lungime] ← 1
					lungime ← lungime + 1
				altfel daca c = '(' atunci
						stiva[lungime] ← -1
						lungime ← lungime + 1
					altfel daca c = ')' atunci
							suma ← 0
							cat-timp stiva[lungime-1] ≠ -1 executa
								suma ← suma + stiva[lungime-1]
								lungime ← lungime - 1
							sf-cat-timp
							stiva[lungime-1] ← suma
						altfel cifra ← c - '0'
							ultim ← stiva[lungime - 1]
							stiva[lungime-1] ← cifra * ultim
						sf-daca
					sf-daca
				sf-daca
			sf-daca
		sf-daca
	sf-pentru

	masa ← 0
	pentru i ← 1, lungime executa
		masa ← masa + stiva[i]
	sf-pentru

	masaMolecula ← masa
sf_functie
```

Exemple

Date de intrare	Rezultat
formula	

CH(CO2H)3	148
CO2H	45
((CH)2(OH2H)(C(H))O)3	222
COOH	45
(((HH)3)3)3	54

Cod sursă C++

```
int masaMolecula(string formula)
{

	int stiva[100];
	int lungime = 0;

	for (int i = 0; i < formula.size(); i++)
	{
		char c = formula.at(i);
		if (c == 'O')
		{
			stiva[lungime++] = 16;
		}
		else if (c == 'C')
			{
				stiva[lungime++] = 12;
			}
			else if (c == 'H')
				{
					stiva[lungime++] = 1;
				}
				else if (c == '(')
					{
						stiva[lungime++] = -1;
					}
					else if (c == ')')
					 {
						int suma = 0;
						while (stiva[lungime - 1] != -1)
						{
							suma += stiva[lungime - 1];
							lungime--;
						}
						stiva[lungime - 1] = suma; //punem suma
					 }
					else {
						int cifra = c - '0';
						int ultim = stiva[lungime - 1];
						stiva[lungime - 1] = cifra * ultim;
						}
	}

	int masa = 0;
```

```
        for (int i = 0; i < lungime; i++)
        {
                masa += stiva[i];
        }
        return masa;
}
```

Cod sursă Pascal

```
Program Molecula;
function masaMolecula(formula:string):integer;
var
    masa, suma, i, cifra, ultim: integer;
    stiva: array [1..100] of integer;
    lungime: integer;
    c: char;
begin
    lungime := 1;
    for i:= 1 to Length(formula) do
    begin
        c := formula[i];
        if c = 'O' then
        begin
            stiva[lungime] := 16;
            lungime := lungime + 1;
        end
        else if c = 'C' then
        begin
            stiva[lungime] := 12;
            lungime := lungime + 1;
        end
        else if c = 'H' then
        begin
            stiva[lungime] := 1;
            lungime := lungime + 1;
        end
        else if c = '(' then
        begin
            stiva[lungime] := -1;
            lungime := lungime + 1;
        end
        else if c = ')' then
        begin
            suma := 0;
            while stiva[lungime - 1] <> -1 do
            begin
                suma := suma + stiva[lungime - 1];
                lungime := lungime - 1;
            end;
            stiva[lungime - 1] := suma;
        end
        else
        begin
            cifra := Ord(c) - Ord('0');
            ultim := stiva[lungime-1];
            ultim := ultim * cifra;
            stiva[lungime-1] := ultim;
```

```
        end;
    end;
    masa := 0;
    for i := 1 to lungime - 1 do
        masa := masa + stiva[i];
    masaMolecula := masa;
end;
begin
  writeln(masaMolecula('COOH'));
  writeln(masaMolecula('CO2H'));
  writeln(masaMolecula('CH(CO2H)3'));
  writeln(masaMolecula('((CH)2(OH2H)(C(H))O)3'));
  writeln(masaMolecula('(((HH)3)3)3'));
end.
```

Problema 4 - Punct de rotație

Enunț

Se deschide un dicționar la întâmplare și se răsfoiește, parcurgând paginile în ordine, de la punctul de deschidere până la final. Pe măsură ce se parcurg paginile, se aleg anumite cuvinte care se adaugă într-un tablou. Când se ajunge la sfârșitul dicționarului, se continuă procesul de la începutul dicționarului, adăugându-se în continuare cuvinte în tablou până când se întâlnește din nou pagina de la care s-a pornit, moment în care s-a creat un tablou *ordonat rotit*. Acesta conține cuvinte care sunt aproape ordonate alfabetic (se începe de pe o poziție aleatorie din alfabet, se ajunge până la finalul acestuia și se reia alfabetul de la început până la punctul de pornire - exclusiv). Un asemenea tablou are un *punct de rotație*: poziția celui mai mic element în ordine alfabetică. De exemplu, punctul de rotație al tabloului *[harta, iarna, inel, joc, auto, carte, desen, ghiocel]* este 5, întrucât pe poziția 5 se află cuvântul *auto,* presupunând că indexarea începe de la 1 (respectiv poziția 4 în codul C++). Se cere găsirea punctului de rotație.

Analiză

Problema poate fi rezolvată folosind o variantă modificată a căutării binare:

- tabloul este sortat până la un punct (punctul de rotație), după care urmează doar elemente care sunt mai mici decât primul element (de pe poziția 1); deci se va căuta prima poziție care conține un element care este mai mic decât elementul de pe poziția 1 din tablou;
- se consideră 2 poziții, *început* și *sfârșit*; se calculează *mijlocul*; dacă elementul de la mijloc este mai mare decât primul element (cel de pe poziția 1), se continuă căutarea în a doua jumătate, altfel se continuă căutarea în prima jumătate;
- în momentul în care *sfârșit* = *început* + 1, căutarea s-a terminat, iar soluția este poziția *sfârșit*.

Specificarea subalgoritmului

Funcție **punctDeRotatie(tablou, n):**
Descriere: caută punctul de rotație într-un tablou de n elemente
Date: tablou – tablou în care fiecare element este un șir de caractere; conținutul tabloului corespunde cerinței din problemă
n – lungimea tabloului
Rezultate: punctul de rotație (poziție în tablou, număr întreg)

Proiectare

```
functie punctDeRotatie(tablou, n) este:
      indexPrim ← 1
      indexUltim ← n
      cattimp (indexPrim < indexUltim) executa
            mijloc ← (indexPrim + indexUltim) div 2
            daca (tablou[1] < tablou[mijloc] atunci
                  indexPrim ← mijloc
```

```
            altfel
                  indexUltim ← mijloc
            sf-daca
            daca (indexPrim + 1 = indexUltim) atunci
                  punctDeRotatie ← indexUltim
            sf-daca
      sf-cattimp
sf-functie
```

Exemple

Date de intrare		Rezultat
tablou	**n**	**Presupunand indexare de la 1**
[“harta”, “iarna”, “inel”,” joc”, “auto”, “carte”,” desen”, “gaina”]	8	5
[“harta”, “iarna”, “inel”,” joc”, “legatura”, “masina”, “pantofi”, “rezultat”, “tango”, “auto”, “carte”, ”curcubeu”, ” desen”, “gaina”]	14	10
["harta", "iarna", "inel", "joc", "lemn", "negru", "primavara", "vale", "vierme", "auto", "carte", "desen", "gaina"]	13	10
["banca", "carte", "curcubeu", "desen", "gaina", "harta", "iarna", "inel","joc", "legatura", "masina", "pantofi", "rezultat", "salsa", "tango", "auto"]	16	16
["tango", "auto", "banca", "carte", "curcubeu", "desen", "gaina", "harta", "iarna", "inel","joc", "legatura", "masina", "pantofi", "rezultat", "salsa"]	16	2

Cod sursă C++

//**OBSERVATIE** – în C++ indexarea începe de la 0, deci rezultatul este cu 1 mai mic decât în exemplele de mai sus

```
int punctDeRotatie(string elemente[], int n)
{

	int indexPrim = 0;
	int indexUltim = n - 1;

	while (indexPrim < indexUltim)
	{
		int mijloc = (indexPrim + indexUltim) / 2;
		if (elemente[0] < elemente[mijloc])
		{
			indexPrim = mijloc;
		}
		else
```

```
            {
                  indexUltim = mijloc;
            }
            if (indexPrim + 1 == indexUltim)
            {
                  return indexUltim;
            }
      }

}
```

Cod sursă Pascal

```
type
T = array[1..100] of string;
function punctDeRotatie (tablou: T; n: integer): integer;
var
    i,indexPrim, indexUltim, mijloc: integer;
    cont: Boolean;
begin
    indexPrim := 1;
    indexUltim := n;
    cont := True;
    while cont do
    begin
        mijloc := (indexPrim + indexUltim) div 2;
        if (tablou[1] < tablou[mijloc]) then
            indexPrim := mijloc
        else
            indexUltim := mijloc;
        if indexPrim + 1 = indexUltim then
        begin
            punctDeRotatie := indexUltim;
            cont := False;
        end;
    end

end;
```

Problema 5 - Găsește duplicatul

Enunț

Problemă suplimentară cu rezolvare în C++/Pascal (cu explicații generale).

Se dă un tablou cu n+1 elemente, în care fiecare element este un număr întreg din intervalul [1,n]. Prin urmare, există cel puțin un element care se repetă. Este posibil ca toate elementele să fie egale. Să se găsească unul dintre elementele care se repetă.

Exemple

Date de intrare		Rezultat
Tablou	**n**	
[1,2,3,4,5,6,2]	7	2
[1,1,1,1,1,1,1]	7	1
[1,2,3,1,2,3,1,2,3,1,2,3]	12	1 sau 2 sau 3
[6,1, 5, 2, 3, 2, 4, 7]	8	2
[1, 1, 1, 1, 7, 7, 7, 7]	8	1 sau 7

Analiză

Varianta 1

- se utilizează 2 cicluri for

Complexitate de spațiu extra (ignorând tabloul de intrare): Θ(1)
Complexitate de timp: $O(n^2)$ (în cel mai rău caz – când ultimele 2 elemente sunt egale, restul unice – se execută ambele cicluri for complet, dar execuția se poate opri și mai repede)

Cod sursă C++

```
int duplicat1(int elemente[], int lungime)
{
	for (int i = 0; i < lungime; i++)
	{
		for (int j = i + 1; j < lungime; j++)
		{
			if (elemente[i] == elemente[j])
			{
				return elemente[i];
			}
		}
	}
}
```

Cod sursă Pascal

type

```
T = array[1..100] of integer;

function duplicat1(tablou: T; n: integer):integer;
var
    i, j, result: integer;
    cont: Boolean;
begin
    i := 1;
    cont := True;
    result := 0;
    while (i <= n) and cont do
    begin
        j := i + 1;
        while (j <= n ) and cont do
        begin
            if (tablou[i] = tablou[j]) then
            begin
                result := tablou[i];
                cont := False;
            end;
        j := j +1;
        end;
    i := i + 1;
    end;
    duplicat1 := result;
end;
```

Varianta 2

- se folosește un tablou auxiliar de n elemente cu valoarea *true* sau *false*;
- inițial, toate elementele din tabloul auxiliar au valoarea *false*; se parcurge tabloul cu numere întregi: pentru fiecare număr *e*, se setează valoarea de pe poziția *e* din tabloul auxiliar la *true*; dacă valoarea este deja *true*, se returnează *e*, fiind un element duplicat.

Complexitate de spațiu extra: Θ(n)
Complexitate de timp: O(n)

Cod sursă C++

```
int duplicat2(int elemente[], int lungime)
{
```

```
	//sunt n+1 elemente din intervalul 1..n; folosim un vector auxiliar de n+1 elemente, nu utilizam pozitia 0
	bool* auxiliar = new bool[lungime];
	for (int i = 0; i < lungime; i++)
	{
		auxiliar[i] = false;
	}
	for (int i = 0; i < lungime; i++)
	{
		int elem = elemente[i];
		if (auxiliar[elem] == true)
		{
			delete[] auxiliar;
			return elem;
		}
		else
		{
			auxiliar[elem] = true;
		}
	}
}
```

Cod sursă Pascal

```
type
T = array[1..100] of integer;
TBool = array[1..100] of Boolean;

function duplicat2(tablou: T; n: integer): integer;
var
    i, elem, result: integer;
    aux: TBool;
    cont : Boolean;
begin
    for i:= 1 to n do
        aux[i] := False;
    cont := True;
    i := 1;
    while (i <= n) and cont do
    begin
        elem := tablou[i];
        if (aux[elem] = true) then
        begin
            cont := False;
            result := elem;
        end
        else
            aux[elem] := True;
        i := i + 1;
    end;
```

```
    duplicat2 := result;
end;
```

Varianta 3

Se folosește o metodă similară cu căutarea binară:

- vectorul de elemente nu se împarte (ca în cazul căutării binare), ci se încearcă reducerea intervalului în care se găsește un element duplicat;
- elementele sunt din intervalul [1...n]; dacă se împarte intervalul în 2 jumătăți, una dintre acestea ar trebui să conțină mai multe elemente decât lungimea intervalului, întrucât cele 2 intervale au împreună lungimea n, dar șirul conține n+1 elemente; elementul care se repetă este în jumătatea respectivă și se poate continua căutarea în acel interval.

Complexitate de spațiu extra: $\Theta(1)$
Complexitate de timp: $\Theta(n*\log_2 n)$

Cod sursă C++

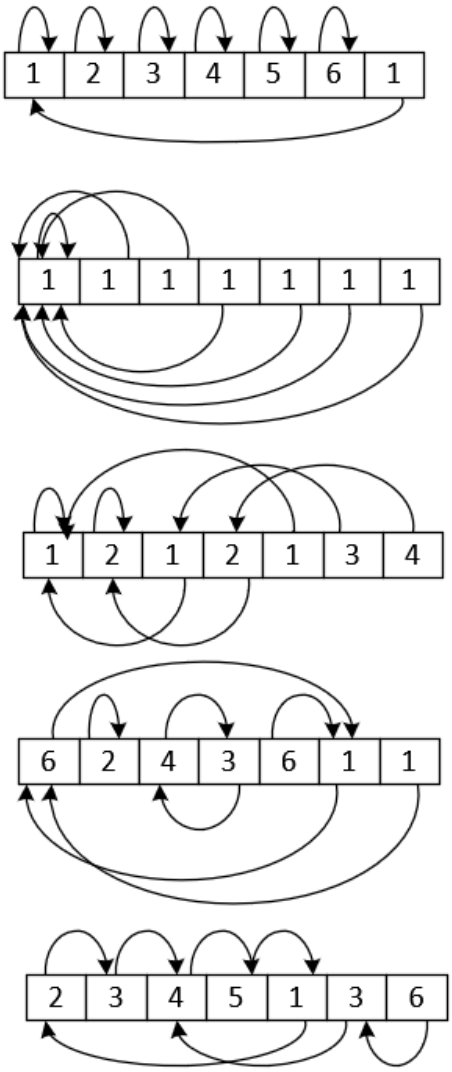
```
int duplicat3(int elemente[], int lungime)
{
	int inceput = 1;
	int sfarsit = lungime - 1;

	while (inceput < sfarsit)
	{
		int mijloc = (inceput + sfarsit) / 2;
		int count = 0;
		for (int i = 0; i < lungime; i++)
		{
			if (elemente[i] >= inceput && elemente[i] <= mijloc)
			{
				count++;
			}
		}
		int lungimeInterval = mijloc - inceput + 1;
		if (lungimeInterval < count)
		{
			sfarsit = mijloc;
		}
		else
		{
			inceput = mijloc + 1;
		}
	}
	return inceput;
}
```

Cod sursă Pascal

```
type
T = array[1..100] of integer;
function duplicat3(tablou: T; n: integer): integer;
var
    inceput, sfarsit, mijloc, result, count, lungimeInterval, i: integer;
begin
    inceput := 1;
    sfarsit := n;
    while (inceput < sfarsit) do
    begin
        mijloc := (inceput + sfarsit) div 2;
        count := 0;
        //numaram cate elemente sunt in intervalul [inceput, mijloc]
        for i := 1 to n do
        begin
            if (tablou[i] >= inceput) and (tablou[i] <= mijloc) then
                count := count + 1;
        end;
        lungimeInterval := mijloc - inceput + 1;
        if (lungimeInterval < count) then
            sfarsit := mijloc
        else
            inceput := mijloc + 1;
    end;
    duplicat3 := inceput;
end;
```

Varianta 4

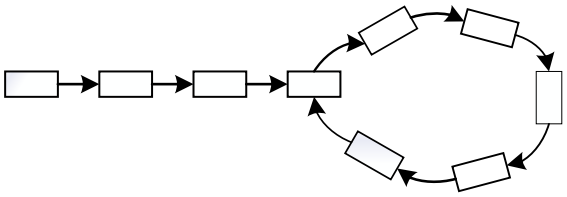
- Întrucât fiecare poziție conține un număr din intervalul [1..n], aceste valori se pot trata ca și cum ar fi poziții din tablou, un fel de pointeri către elementul următor; se presupune că numerotarea pozițiilor începe de la 1.
- exemple:



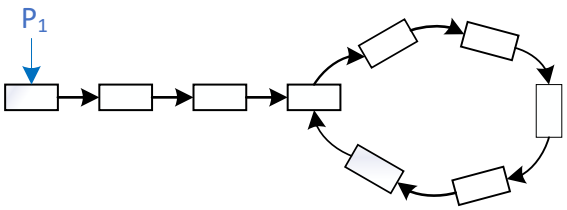
- Dacă ne uităm la desene, putem observa că în anumite poziții intră mai mulți *pointeri*. Și că fiecare poziție în care intră mai mulți pointeri reprezintă un element care se repetă.
- Dacă folosim aceste legături "fictive", putem parcurge tabloul ca și cum ar fi o listă înlănțuită. Considerând ultimul exemplu de mai sus, dacă pornim de pildă de la poziția 1, trecem prin următoarele elemente/poziții:
    - poz 1 -> poz 2 -> poz 3 -> poz 4 -> poz 5 -> poz 1 -> poz 2 -> poz 3 … etc.

  - Sau dacă pornim de la poziția 6 vom avea: poz 6 - > poz 3 -> poz 4 -> poz 5 -> poz 1 -> poz 2 -> poz 3 -> poz 4 … etc.
  - Sau dacă pornim de la poziția 7 vom avea: poz 7 - > poz 6 -> poz 3 -> poz 4 -> poz 5 -> poz 1 -> poz 2 -> poz 3 -> poz 4 … etc.
- De la care poziție ar trebui să începem parcurgerea?
- Primul element dintr-o listă ar trebui să fie un element spre care nu *pointează* niciun alt element. Avem n+1 poziții, dar elementele sunt până la n, nu există niciun element care să *pointeze* spre ultimul element din listă, ar trebui să începem de acolo parcurgerea.
- Pentru exemplele de mai sus, aceste parcurgeri ar fi:
  - poz 7 -> poz 1 -> poz 1 -> poz 1 …
  - poz 7 -> poz 1 -> poz 1 -> poz 1 ...
  - poz 7 -> poz 4 ->poz 2 -> poz 2 -> poz 2 ...
  - poz 7 -> poz 1 -> poz 6 -> poz 1 -> poz 6 …
  - poz 7 -> poz 6 -> poz 3 -> poz 4 -> poz 5 -> poz 1 -> poz 2 -> poz 3 -> poz 4 -> poz 5 …
- Putem observa că fiecare parcurgere la un moment dat se termină înt-un ciclu, iar prima poziție care se repetă în parcurgere (adică începutul ciclului) reprezintă elementul care se repetă (elem 1, elem 1, elem 2, elem 6, elem 3 în exemplele de mai sus)
- Cum găsim începutul ciclului?
  - Dacă am ști că lungimea ciclului (numărul de elemente) este L, am putea folosi metoda următoare:
    - Pornim în parcurgere (de la ultimul element) și facem L pași.
    - După ce am făcut L pași, pornim cu încă un *pointer* de la început și mergem în paralel.
    - Când cei 2 pointeri se întâlnesc, am găsit începutul ciclului.

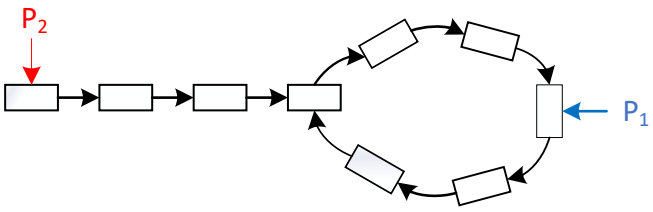
De exemplu, considerăm lista din figura de mai jos, pentru care știm că lungimea ciclului este 6.



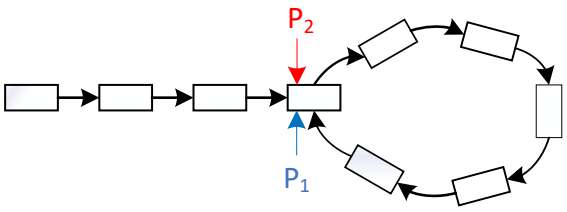
Pornim cu un pointer P1 de la primul element:



Din moment ce știm că lungimea ciclului este 6, facem 6 pași cu pointerul P1 (deci de 6 ori mergem la nodul următor). După care luăm un pointer P2 pe care îl setăm la începutul listei:



Mai departe mergem în paralel cu cei 2 pointeri, până când ei *pointează* spre același element. Acel element e începutul ciclului (în figura de mai sus, sunt necesari 3 pași):



- Metoda de mai sus se bazează pe faptul că știm lungimea ciclului. Cum găsim această lungime?
    - Dacă la un moment dat știm că suntem în interiorul ciclului (nu contează unde), putem reține elementul curent și continua deplasarea până ajungem înapoi la acel element. Pe parcursul deplasării numărăm câți pași facem. Acest număr este lungimea ciclului.
- Cum ne asigurăm că suntem în interiorul ciclului?
    - Există n valori posibile pentru elementele tabloului, deci lungimea maximă a unui ciclu este n. Dacă ne deplasăm n+1 pași pornind de la începutul listei (adică ultima poziție), știm sigur că suntem în listă.

- Deci algoritmul pe scurt:
    - Pornim de la ultima poziție și ne deplasăm n+1 pași, ca să ne asigurăm că suntem în interiorul ciclului.
    - Reținem elementul curent, pornim un contor și continuăm deplasarea, până ajungem înapoi la elementul curent.
    - Când știm lungimea ciclului, pornim o parcurgere nouă de la început, facem *lungime* pași, după care pornim cu un al 2-lea pointer și mergem în paralel, până când cei 2 pointeri sunt pe aceeași poziție. Această poziție reprezintă începutul ciclului și unul dintre elementele care se repetă.

Implementare C++

```
int duplicat4(int elemente[], int lungime)
{

	int curent = lungime; //pozitia curenta din tablou
	for (int i = 0; i < lungime; i++)
	{
		curent = elemente[curent - 1]; //ne trebuie -1, pentru ca in C++ pozitiile se
numara de la 0 nu de la 1
	}
	//acum curent este in ciclu
	int elemDinCiclu = curent;
	int lungimeCiclu = 1;
	curent = elemente[curent - 1];
	while (curent != elemDinCiclu)
	{
		lungimeCiclu++;
		curent = elemente[curent - 1];
	}
	//lungime e lungimea ciclului.
	int pointerNou = lungime; //pornim cu un pointer nou de la inceput
	for (int i = 0; i < lungimeCiclu; i++)
	{
		pointerNou = elemente[pointerNou - 1];
	}
	int pointerNou2 = lungime;
	//acum mergem in paralel cu pointeri
	while (pointerNou != pointerNou2)
	{
		pointerNou = elemente[pointerNou - 1];
		pointerNou2 = elemente[pointerNou2 - 1];
	}
	return pointerNou;

}
```

Complexitate de spațiu extra: Θ(1)
Complexitate de timp: Θ(n)

Implementare Pascal

```
type
T = array[1..100] of integer;

function duplicat4(tablou: T; n: integer): integer;
var
    elemDinCiclu, lungimeCiclu, pointerNou, pointerNou2, curent, i:
integer;
begin
```

```
    curent := n; //pornim de la ultima pozitie
    //de n+1 ori mergem la pozitia spre care "pointeaza" valoarea de pe
pozitia curenta
    for i := 1 to n+1 do
        curent := tablou[curent];
    //stim sigur ca suntem in ciclu. Calculam lungimea ciclului
    elemDinCiclu := curent;
    lungimeCiclu := 1;
    curent := tablou[curent];
    while (curent <> elemDinCiclu) do
    begin
        lungimeCiclu := lungimeCiclu + 1;
        curent := tablou[curent];
    end;
    //pornim de la capatul tabloului (adica inceputul listei) si facem
lungimeCiclu pasi
    pointerNou := n;
    for i:=1 to lungimeCiclu do
        pointerNou := tablou[pointerNou];
    //pornim cu inca un pointer
    pointerNou2 := n;
    while (pointerNou <> pointerNou2) do
    begin
        pointerNou := tablou[pointerNou];
        pointerNou2 := tablou[pointerNou2];
    end;
    duplicat4 := pointerNou;
end;
```

Întrebări grilă

Întrebarea 1a.

Considerați subalgoritmul următor:

```
void s1(int tablou[], int n)
{
	for (int i = 0; i < 5; i++)
	{
		for (int j = 0; j < n - 1; j++)
		{
			if (tablou[j] > tablou[j + 1])
			{
				int tmp = tablou[j];
				tablou[j] = tablou[j + 1];
				tablou[j + 1] = tmp;
			}
		}
	}
}
```

Care dintre următoarele afirmații despre subalgoritmul s1 este adevărată?

a. Subalgoritmul s1 sortează un tablou indiferent de lungimea tabloului.
b. Subalgoritmul s1 sortează un tablou dacă are lungimea 5.
c. Subalgoritmul s1 sortează un tablou dacă are lungimea mai mică decât 5.
d. Subalgoritmul s1 sortează un tablou dacă are lungimea mai mare decât 5.

Întrebarea 1b.

Considerați același subalgoritm s1 (de la întrebarea 1a). Care dintre următoarele afirmații este adevărată?

a) După rularea subalgoritmului s1, indiferent de lungimea tabloului, pe ultima poziție va fi cel mai mare element din tablou.
b) După rularea subalgoritmului s1, indiferent de lungimea tabloului, pe ultima poziție va fi cel mai mic element din tablou.
c) După rularea subalgoritmului s1, indiferent de lungimea tabloului, pe prima poziție va fi cel mai mic element din tablou.
d) După rularea subalgoritmului s1, indiferent de lungimea tabloului, pe prima poziție va fi cel mai mare element din tablou.

Întrebarea 2a.

Considerați subalgoritmul următor:

```
int s2(int tablou[], int n, int p, int c) {
	if (p == n) {
		return c;
	}
	else if (c > tablou[p]) {
		return s2(tablou, n, p + 1, c);
	}
	else if (c < tablou[p]) {
		return s2(tablou, n, p + 1, tablou[p]);
	}
	return 0;
}
```

Ce va returna subalgoritmul s2, dacă îl apelăm cu *tablou = [1, 2, 3, 4, 5, 5]*, *n = 6, p = 0, c = -1*?

a. -1
b. 0
c. 5
d. 6

Întrebarea 2b.

Considerați același subalgoritm s2 (de la întrebarea 2b). De câte ori se execută comparația `c < tablou[p]` dacă apelăm s2 cu *tablou = [-1, 7, 3, 9, 2, 11], n = 6, p = 1, c = -1*?

a. 0
b. 2
c. 3
d. 5

Întrebarea 3.

Considerați problema verificării existenței unui element *e* într-un tablou *tablou* cu lungimea *n*. Care dintre următorii subalgoritmi reprezintă rezolvări corecte pentru această problemă?

a. Subalgoritmul a1

```
bool a1(int tablou[], int n, int e)
{
	bool g = false;
	int i = 0;
	while (i < n)
	{
		if (tablou[i] == e)
		{
			g = true;
		}
		else
		{
			g = false;
		}
		i = i + 1;
	}
	return g;
}
```

b. Subalgoritmul a2

```
bool a2(int tablou[], int n, int e)
{
	bool g = false;
	int i = 0;
	while (i < n and !g)
	{
		if (tablou[i] == e)
		{
			g = true;
		}
		else
		{
			g = false;
		}
		i = i + 1;
	}
	return g;
}
```

c. Subalgoritmul a3

```
bool a3(int tablou[], int n, int e)
{
	int c = 0;
	for (int i = 0; i < n; i++)
```

```
        {
                if (tablou[i] == e)
                {
                        c++;
                }
                else
                {
                        c--;
                }
        }
        return -1 * n != c;
}
```

d. Subalgoritmul a4

```
bool a4(int tablou[], int n, int e)
{
        int i = 0;
        bool g = false;
        while (i < n)
        {
                if (tablou[i] < e + 1 && tablou[i] % e == 0)
                {
                        g = true;
                }
                i = i + 1;
        }
        return g;
}
```

Întrebarea 4.

Se consideră problema: se dă un cuvânt (reprezentat ca un tablou de caractere de lungime n) format doar din literele mici ale alfabetului englez; verificați dacă se poate forma un palindrom rearanjând literele cuvântului. Se propune algoritmul de mai jos pentru rezolvarea problemei:

```
bool palindrom(char tablou[], int n)
{
	int f[26];
	for (int i = 0; i < 26; i++)
	{
		f[i] = 0;
	}
	for (int i = 0; i < n; i++)
	{
		f[tablou[i] - 'a'] ++;
	}
	bool g = true;
	int m = 0;
	int i = 0;
	while (i < 26 && g)
	{
		if (_____________________________________)
		{
			g = false;
		}
		else if (f[i] % 2 == 0)
		{
			i = i + 1;
		}
		else
		{
			m++;
			i = i + 1;
		}
	}
	return g;
}
```

Care dintre următoarele condiții ar trebui adăugată pe linia marcată pentru a obține o soluție corectă?

a. f[i] % 2 == 1
b. f[i] % 2 == 1 && m >= 1
c. f[i]%2 == 1 && m== 1
d. g == true && f[i] % 2 == 1

Întrebarea 5.

Încercuiți expresia calculată de secvența de cod:

```
int n;
float s = 0, p = 1;
do
{
	cout << "n: ";
	cin >> n;
} while (n <= 0);
for (int i = 1; i <= n; i++)
{
	p *= i;
	s += 1 / p;
}
```

a. $\frac{1}{1^2}+\frac{1}{2^2}+\cdots+\frac{1}{n^2}$
b. $\frac{1}{1!}+\frac{1}{2!}+\cdots+\frac{1}{n!}$
c. $\frac{1}{2^1}+\frac{1}{2^2}+\cdots+\frac{1}{2^n}$
d. $\frac{1}{n^2}+\frac{1}{n^2}+\cdots+\frac{1}{n^n}$

Întrebarea 6.

Ce valoare afișează secvența de cod:

```
int b = 0, c = 0;
for (int i = 0; i < 10; i++)
{
        c += i;
        b += ++c - 1;
}
cout << b;
```

a. 45
b. 210
c. 120
d. 0

Întrebarea 7.

Se consideră subprogramul f cu definiţia de mai jos. Ce valoare are f(0,100)?

```
int f(int x, int y)
{
	if (x < y - 1)
		return 1 + f(x + 1, y - 1);
	else
		return 0;
}
```

a. 48
b. 49
c. 50
d. 51

Răspunsuri grilă:

1a. - b, c
1b. - a
2a. - b
2b. - c
3 - b, c
4 - b, c
5 - b
6 - b
7 - c