TABLOURI UNIDIMENSIONALE

Problema 1

Enunț

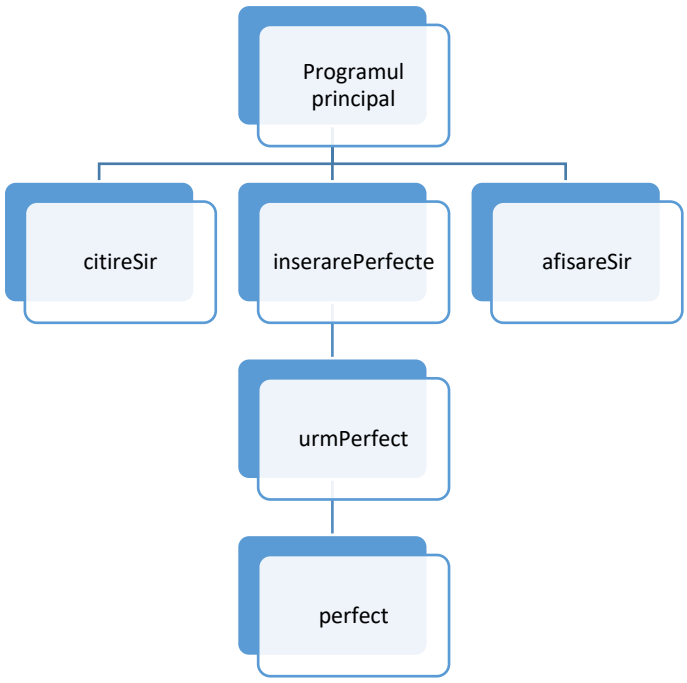
Se citesc *n* numere naturale (valoarea lui *n* este citită) şi se memorează într-un şir. Să se insereze, după fiecare număr din șir următorul număr perfect. Un număr perfect este un număr natural egal cu suma divizorilor săi pozitivi proprii (toți divizorii pozitivi, cu excepția numărului însuși).

Exemplu:
Dat fiind șirul: 100 18 30 5 5496
Șirul rezultat va fi: 100 496 18 28 30 496 5 6 5496 8128.

Se cere să se utilizeze subprograme, care să comunice între ele şi cu programul principal prin parametri.

Analiză

Identificarea subalgoritmilor



Specificarea subalgoritmilor

- Subalgoritmul **citireSir**(a,n)
  Descriere: Citește șirul de numere naturale *a*, de lungime *n.*
  Date: -
  Rezultate: *a, n – a* vector de numere naturale de lungime *n*.
  $a=(a_i \mid i=1..n, a_i \in N), n \in N$

- Subalgoritmul **inserarePerfecte**(a,n)
  Descriere: Inserează în vectorul *a* de lungime *n* după fiecare element numărul perfect mai mare decât el.
  Date: *a, n*, $a=(a_i \mid i=1..n, a_i \in N), n \in N$
  Rezultate: *a, n*
  a=(a_i *|i=1..n,* a_i *∈N), n∈N, a va avea inserat dupa fiecare element urmatorul numar perfect mai mare decat el*

- Subalgoritmul **inserareElem**(a,n,pos,elem)
  Descriere: Inserează elementul *elem* la poziția *pos* în vectorul *a* de lungime *n.*
  Date: *a,n, pos, elem*
  a - vector de lungime *n*
  $n \in N$,
  $pos \in N, 0<=pos<=n$
  $elem \in N$
  Rezultate: *a, n*
  $a=(a_i, i=1..n, a$ *va contine elementul elem la pozitia pos),* $n \in N$

- Subalgoritmul **afisareSir**(a,n)
  Descriere: Afișează elementele din vectorul *a* de lungime *n*.
  Date: *a, n – a* vector de lungime *n*, $n \in N$.
  $a=(a_i \mid i=1..n, a_i \in N)$,
  Rezultate: -
  se afișează elementele din vectorul *a*.

- Funcția **perfect**(x)
  Descriere: Verifică dacă numarul *x* este perfect.
  Date: *x* – numar natural, $x \in N$
  Rezultate: returnează *true* dacă numarul *x* e perfect, *false* altfel.

- Funcția **urmPerfect**(x)
  Descriere: Găseste următorul număr perfect mai mare decat *x*.
  Date: x – numar natural, $x \in N$
  Rezultate: returnează următorul număr perfect mai mare decat *x.*

Implementare C++

```
#include <iostream>

using namespace std;

bool perfect(int n) {
	if (n == 0)
		return false;
	int sum = 0;
	for (int i = 1; i <= (int)n / 2; i++)
		if (n % i == 0)
			sum += i;
	return (n == sum);
}

int urmPerfect(int n) {
	int x = n + 1;
	while (!perfect(x))
		x++;
	return x;
}

void inserareNumerePerfecte(int x[100], int n) {
	for (int i = n - 1; i >= 0; i--)
	{
		x[2 * i + 1] = urmPerfect(x[i]);
		x[2 * i] = x[i];
	}
}

void citireSir(int x[100], int& n) {
	cout << "Dimensiunea sirului: ";
	cin >> n;
	for (int i = 0; i < n; i++) {
		cout << "x[" << i << "]=";
		cin >> x[i];
	}
}

void tiparireSir(int x[100], int n) {
	for (int i = 0; i < n; i++) {
		cout << x[i] << " ";
	}
}

int main() {
	int x[100];
	int n = 5;
	citireSir(x, n);
	inserareNumerePerfecte(x, n);
	tiparireSir(x, 2 * n);

	return 0;
}
```

Exemple

Date de intrare		Rezultate
n	sir	
5	100, 18, 30, 5, 5496	100, 496, 18, 28, 30, 496, 5, 6, 5496, 8128
4	10, 850, 1000, 2	10, 28, 850, 8128, 1000, 8128, 2, 6
6	-5, 1, 10, 10, 2, 2	-5, 6, 1, 6, 10, 28, 10, 28, 2, 6, 2, 6
1	5000	5000, 8128
5	1, 10, 1000, -1, 400	1, 6, 10, 28, 1000, 8128, -1, 6, 400, 496

Problema 2

Enunț

Institutul Meteorologic Cluj are nevoie de o aplicație care să premită menținerea evidenței tuturor temperaturilor medii anuale (în grade Celsius), începând de la un an dat, până în anul precedent celui curent (până în 2024, în acest caz). Aplicația trebuie să permită:

- Introducerea unui an de început și apoi a temperaturilor medii anuale, până în anul precedent celui curent (2024). De exemplu, dacă anul de început este 2008, se vor reține 17 temperaturi medii (pentru 2008, 2009, 2010, ... , 2024).
- Afișarea celei mai lungi perioade și a valorilor temperaturilor asociate în care temperaturile au crescut în continuu.
- Afișarea temperaturilor maxime, din fiecare perioadă în care temperaturile au crescut și apoi au scăzut.

Exemplu:

Pentru an curent 2025, an de început 2008 și temperaturile medii anuale:

15.2, 12.1, 10.8, 11, 14.6, 15.8, 10, 11.3, 12.1, 12.7, 13.1, 13, 12.8, 12.6, 12, 11.8, 11.5

Cea mai lungă perioadă în care temperaturile au crescut în continuu și valorile temperaturilor:

2014 - 2018

10, 11.3, 12.1, 12.7, 13.1

Toate perioadele în care temperaturile au crescut, cu maximele lor (se cer doar maximele, dar sunt exemplificate şi perioadele pentru o mai bună înţelegere):

10.8, 11, 14.6, 15.8 -> max: 15.8

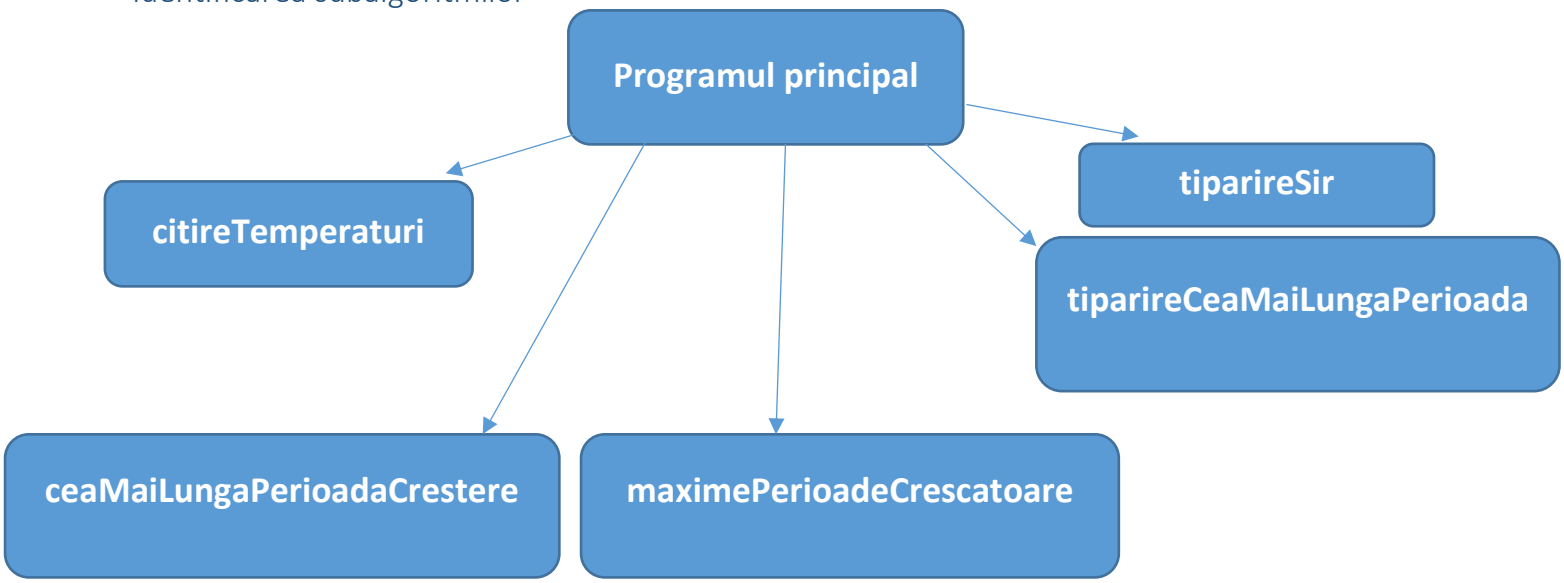
10, 11.3, 12.1, 12.7, 13.1 -> max: 13.1

Temperatura maximă din toată perioada indicată: 15.8

Se cere să se utilizeze subprograme, care să comunice între ele şi cu programul principal prin parametri. Fiecare subprogram trebuie specificat.

Analiză

Identificarea subalgoritmilor



Specificarea subalgoritmilor

- Subalgoritmul **citireTemperaturi(sirTemperaturi, lungimeSir)**:
  Descriere: Cunoscându-se anul curent, citește șirul acestor temperaturi și calculează lungimea șirului acestuia.
  Date: -
  Rezultate: - *sirTemperaturi* – șir de numere reale, reprezentând fiecare câte o temperatură anuală
  sirTemperaturi = $(t_i \in \boldsymbol{R} | i=1..lungimeSir, lungimeSir \in \boldsymbol{N})$
  - *lungimeSir* $\in \boldsymbol{N}$, lungimea șirului temperaturilor.

- Subalgoritmul **ceaMaiLungaPerioadaCrestere (sirTemperaturi, lungimeSir, pozitieStart, lungimeSecventa)**:

Descriere: Identifică cea mai lungă perioadă cu temperaturi creascătoare.
Date: - *sirTemperaturi* – șir de numere reale, reprezentând fiecare câte o temperatură anuală *sirTemperaturi = (t_i ∈ **R**|i=1..lungimeSir, lungimeSir ∈ **N**)*
- *lungimeSir ∈ $\boldsymbol{N}$*, lungimea șirului temperaturilor.
Rezultate: - *pozitieStart ∈ $\boldsymbol{N}$* – poziția de început a celei mai lungi secvențe creascătoare
- *lungimeSecventa ∈ $\boldsymbol{N}$*, lungimea celei mai lungi secvente crescătoare începand de la poziția dată.

- Subalgoritmul **maximePerioadeCrescatoare(sirTemperaturi, lungimeSir, sirMaxime, lungimeSirMaxime)**:
Descriere: Identifică maximele tuturor secvențelor de temperaturi crescătoare.
Date: - *sirTemperaturi* – șir de numere reale, reprezentând fiecare câte o temperatură anuală *sirTemperaturi = (t_i ∈ **R**|i=1..lungimeSir, lungimeSir ∈ **N**)*
- *lungimeSir ∈ $\boldsymbol{N}$*, lungimea șirului temperaturilor.
Rezultate: - *sirMaxime* – șir de numere reale, conținând maximele tuturor secvențelor de temperaturi crescătoare *sirMaxime = (t_i ∈ **R**|i=1..lungimeSirMaxime, lungimeSirMaxime ∈ **N**)*
- *lungimeSirMaxime ∈ $\boldsymbol{N}$*, lungimea șirului conținând elementele maxime ale secvențelor crescătoare.

- Subalgoritmul **tiparireSecventaDeLaPozitie (sir, lungimeSir, lungimeSecventa, pozitieInceput)**:
Descriere: Tipărește secvența din sirul dat, care începe de la poziția dată, având lungimea dată.
Date: - *sir* – șir de numere reale, reprezentând fiecare câte o temperatură anuală *sir = (t_i ∈ **R**|i=1..lungimeSir, lungimeSir ∈ **N**)*
- *lungimeSir ∈ $\boldsymbol{N}$*, lungimea șirului.
- *lungimeSecventa ∈ $\boldsymbol{N}$,* lungimea secvenței de numere care trebuie tipărite.
- *pozitieInceput ∈ $\boldsymbol{N}$,* poziţia începând de la care trebuie tipărite numerele din şirul dat.
Rezultate: se afiseaza elementele subsecvenței din șirul dat, care începe de la pozitia *pozitieInceput*, și are lungimea *lungimeSecventa*.

- Subalgoritmul **tiparireCeaMaiLungaPerioada (sirTemperaturi, lungimeSir, pozitieStart, lungimePerioada)**:
Descriere: Tipareste cea mai lunga perioada in care temperaturile au crescut in continuu, precum si valorile temperaturilor asociate.
Date: - *sirTemperaturi* – șir de numere reale, reprezentând fiecare câte o temperatură anuală *sirTemperaturi = (t_i ∈ **R**|i=1..lungimeSir, lungimeSir ∈ **N**)*
- *lungimeSir ∈ $\boldsymbol{N}$*, lungimea șirului.
- *lungimePerioada ∈ $\boldsymbol{N}$,* lungimea secvenței de numere care trebuie tipărite.
- *pozitieStart ∈ $\boldsymbol{N}$,* poziţia începând de la care trebuie tipărite numerele din şirul dat.

Rezultate: se afisează cea mai lungă perioadă în care temperaturile au crescut în continuu, precum și valorile temperaturilor asociate.

- Subalgoritmul **tiparireSir**(sirTemperaturi, lungimeSir)
  Descriere: Afișează elementele din vectorul *sirTemperaturi* de lungime *lungimeSir*.
  Date: *sirTemperaturi, lungimeSir* – *sirTemperaturi* vector de lungime *lungimeSir*, *lungimeSir* ∈ N.
  $sirTemperaturi = (t_i \in \mathbf{R} | i=1..lungimeSir, lungimeSir \in \mathbf{N})$
  Rezultate: -
  se afișează elementele din vectorul *sirTemperaturi*.

Implementare

```
#include <iostream>

using namespace std;

const int anCurent = 2025;

void citireTemperaturi(double x[50], int& n) {
	int anInceput = 0;
	cout << "An inceput: ";
	cin >> anInceput;
	n = anCurent - anInceput;
	for (int i = 0; i < n; i++) {
		cout << "x[" << i << "]=";
		cin >> x[i];
	}
}

void tiparireSir(double x[50], int n) {
	for (int i = 0; i < n; i++) {
		cout << x[i] << " ";
	}
}

void tiparireCeaMaiLungaPerioada(double x[50], int n, int idxStart, int l) {
	int anInceput = anCurent - n + idxStart;
	int anFinal = anInceput + l - 1;
	cout << "Perioada: " << anInceput << " - " << anFinal << endl;
	for (int i = idxStart; i < idxStart + l; i++) {
		cout << x[i] << " ";
	}
}

void ceaMaiLungaPerioadaCrestere(double x[50], int n, int& start, int &l) {
	int idxStart = 0, idxEnd = 0, idxStartMax = 0, idxEndMax = 0;
	int lung = 0, lungMax = 0;

	int i = 0;
	while (i < n - 1) {
		idxStart = i;
		lung = 1;
		int j = i + 1;
		while (j < n && x[j] > x[j - 1])
			j++;
```

```
			idxEnd = j - 1;
			lung = idxEnd - idxStart + 1;

			if (lung > lungMax)
			{
				lungMax = lung;
				idxStartMax = idxStart;
				idxEndMax = idxEnd;
			}

			i = j;
		}

		l = lungMax;
		start = idxStartMax;
}

void maximePerioadeCrescatoare(double x[50], int n, double maxime[50], int& l) {
		int k = 0;

		int i = 0;
		while (i < n - 1) {
			int j = i + 1;
			while (j < n && x[j] > x[j - 1])
				j++;
			if (j - i >= 2) {
				double max = x[j - 1];
				maxime[k] = max;
				k++;
			}

			i = j;
		}
		l = k;
}

int main() {
		double x[50]; int n = 0;
		citireTemperaturi(x, n);

		int l = 0, idxStart = 0;
		ceaMaiLungaPerioadaCrestere(x, n, idxStart, l);
		tiparireCeaMaiLungaPerioada(x, n, idxStart, l);
		cout << endl;

		double maxime[50]; int lMax = 0;
		maximePerioadeCrescatoare(x, n, maxime, lMax);
		tiparireSir(maxime, lMax);

		return 0;
}
```

Exemple

Date de intrare		Rezultate
An inceput	Sir temperaturi	
2008	15.2, 12.1, 10.8, 11, 14.6, 15.8, 10, 11.3, 12.1, 12.7, 13.1, 13, 12.8, 12.6, 12, 11.8, 11.5	Cea mai lunga perioada in care temperaturile au crescut in continuu este: 2005 – 2009. 10, 11.3, 12.1, 12.7, 13.1 Temperaturile maxime, din fiecare perioada in care temperaturile au crescut si apoi au scazut: 15.8, 13.1.
2019	14, 15, 13, 14, 15, 16	Cea mai lunga perioada in care temperaturile au crescut in continuu este: 2021 – 2024. 13, 14, 15, 16 Temperaturile maxime, din fiecare perioada in care temperaturile au crescut si apoi au scazut: 15, 16.
2017	11, 12, 12.3, 14.8, 13.1, 14, 14.5, 14.2	Cea mai lunga perioada in care temperaturile au crescut in continuu este: 2017 – 2020. 11, 12, 12.3, 14.8 Temperaturile maxime, din fiecare perioada in care temperaturile au crescut si apoi au scazut: 14.8, 14.5.
2022	15, 15.5, 16	Cea mai lunga perioada in care temperaturile au crescut in continuu este: 2022 – 2024. 15, 15.5, 16 Temperaturile maxime, din fiecare perioada in care temperaturile au crescut si apoi au scazut: 16.

Problema 3

Enunț

Fie *a* un șir de n numere naturale distincte *(a_1,a_2, …, a_n)* ordonat crescător și un numar natural *S*. Scrieți un subalgoritm care determină numărul perechilor *(a_i, a_j)* care îndeplinesc condiția *$S=a_i+a_j$ și $i<j$*.

Exemple:

A = (2, 4, 5, 8, 10, 11), S = 12 => 2 perechi: (2,10) și (4,8)

a = (1, 3, 4, 6, 7, 9, 10), S = 10 => 3 perechi: (1, 9), (3, 7), (4, 6)

a = (4, 6, 8, 10), S = 50 => 0 perechi

Analiză

Varianta 1: O variantă de soluție este parcurgerea șirului a cu două contoare i,j (i=1,2, …, n-1 și j=i+1,i+2, …, n) și verificarea dacă perechea *(a_i,a_j)* îndeplinește condiția *$a_i+a_j=S$*. Această soluție are complexitatea $O(n^2)$ și nu ține cont de condițiile din enunț care precizează că șirul este ordonat crescător și că elementele șirului sunt distincte.

```
int varianta1(int a[100], int n, int s) {
	int count = 0;
	for (int i = 0; i < n - 1; i++)
		for (int j = i + 1; j < n; j++)
			if (a[i] + a[j] == s)
				count++;
	return count;
}
```

Varianta 2: O altă soluție, care ține cont de faptul că șirul *a* este ordonat crescător este să folosim căutarea binară. Parcurgem șirul a, începând cu elementul cel mai mic. Pentru fiecare element și verificăm, folosind căutarea binară, dacă și valoarea *$S-a_i$* apare în șirul a printre elementele *a_{i+1}, …, a_n*. Complexitatea acestei soluții este O(n*log n).

```
bool cautareBinaraRec(int numarDeCautat, int a[100], int n, int st, int dr) {
	if (dr < st)
		return false;
	int mij = (st + dr) / 2;
	if (numarDeCautat == a[mij])
		return true;
	if (numarDeCautat < a[mij])
		return cautareBinaraRec(numarDeCautat, a, n, st, mij - 1);
	return cautareBinaraRec(numarDeCautat, a, n, mij + 1, dr);
}

bool cautareBinara(int numarDeCautat, int a[100], int n, int pozInceput) {
```

```
	return cautareBinaraRec(numarDeCautat, a, n, pozInceput, n - 1);
}

int varianta2(int a[100], int n, int s) {
	int count = 0;
	for (int i = 0; i < n - 1; i++) {
		int numarDeCautat = s - a[i];
		bool gasit = cautareBinara(numarDeCautat, a, n, i + 1);
		if (gasit)
			count++;
	}
	return count;
}
```

Varianta 3: O altă soluție posibilă, care ține cont de condițiile din enunț (șirul a este ordonat crescător și elementele șirului sunt numere distincte) și încearcă să obțină numărul perechilor care satisfac condiția, este să parcurgem șirul o singură dată, dar să folosim doi indecsi: un index *st* parcurge șirul de la pozitia 1 spre n, iar un index *dr* de la poziția n spre 1. Pentru fiecare pereche *(st,dr), st<dr*, astfel obținută, verificăm condiția ca $a_{st}+a_{dr}=S$.

- Dacă perechea îndeplinește condiția, o numărăm, incrementăm *st* și decrementăm *dr* pentru a căuta o altă pereche care îndeplinește condiția.
- Dacă $a_{st}+a_{dr}>S$ înseamnă că trebuie să scădem valoarea sumei $a_{st}+a_{dr}$ și încercăm cu elementul aflat pe poziția *dr-1*
- Dacă $a_{st}+a_{dr}<S$ înseamnă că trebuie să mărim valoarea sumei $a_{st}+a_{dr}$ și încercăm cu elementul aflat pe pozitia *st+1*.

```
int varianta3(int a[100], int n, int s) {
	int count = 0;
	int st = 0;
	int dr = n - 1;
	while (st < dr) {
		if (a[st] + a[dr] == s) {
			count++;
			st++;
			dr--;
		}
		else if (a[st] + a[dr] < s)
			st++;
		else
			dr--;
	}
	return count;
}
```

Exerciții grilă

1. Se consideră algoritmul *Calculeaza(x,n)*, unde *n* este un număr natural ($1 \le n \le 10^3$) si *x* este un vector cu n elemente numere întregi (*x[1], x[2], ..., x[n]*), unde $-10^4 \le x[i] \le 10^4$, pentru *i = 1, 2, ..., n*:

```
Algorithm Calculeaza(x, n)
    s ← x[1]
    a ← x[1]
    For i ← 2, n execute
      If a + x[i] > x[i] then
            a ← a + x[i]
      Else
            a ← x[i]
      EndIf
      If a > s then
            s ← a
      EndIf
    EndFor
    Return s
EndAlgorithm
```

Care dintre următoarele afirmații sunt adevarate?
A. Pentru x = [−2, 1,−3, 4,−1, 2, 1,−5, 4], rezultatul returnat de *Calculeaza(x,9)* este 6;
B. Pentru x = [1,−2, 3, 5,−1], rezultatul returnat de *Calculeaza(x,5)* este 8;
C. Pentru x = [−1,−2,−3,−4], rezultatul returnat de *Calculeaza(x,4)* este −1;
D. Pentru x = [2, 4,−1, 3], rezultatul returnat de *Calculeaza(x,4)* este 6.

Discuții:
Acesta este **algoritmul lui Kadane** (Joseph Born Kadane) – calculează suma maxima a unei subsecvențe (continue) dintr-un șir, în timp liniar (O(n)).

Ideea algoritmului: la fiecare poziție i, alegem între a extinde subsecvența curentă sau a începe o subsecvența nouă la elementul curent, alegând varianta care duce la suma cea mai mare. La fiecare pas se ține evidenta sumei maxime întalnite până atunci.

Fie:
-a[i] – suma maxima a subsecvenței care se termina la poziția i

-s – suma maximă de până acum

Formula de recurență:
a[1] = x[1]
a[i] = max(x[i], a[i-1] + x[i])

s = max(s, a)

O altă variantă pentru a calcula această sumă este prin “brute force”, pentru fiecare indice posibil i, calculăm sumele pentru toate subsecvențele [i, j] posibile și reținem mereu suma maximă. Această variantă are complexitate $O(n^2)$.

R: A, B, C

2. Se consideră subalgoritmul ***h(n)***, unde ***n*** este un număr natural ($1 \leq n \leq 10000$) și ***A*** este un vector cu ***n*** elemente numere întregi.

```
Algorithm h(A, n):
    If n = 0 then
      return 0;
    EndIf
    If n % 2 = 0 then
      return h(A,n-1) + A[n];
    EndIf
    Return h(A,n-1) - A[n];
EndAlgorithm
```

Algoritmul calculează:

A. Numărul elementelor pare din vectorul A

B. Suma elementelor de pe pozițiile pare din vectorul A

C. Diferența elementelor de pe pozițiile impare din vectorul A

D. Diferența dintre suma elementelor pare din vector și suma elementelor impare din vectorul A

E. Diferența dintre suma elementelor de pe poziții pare și suma elementelor de pe pozițiile impare din vectorul A

F. Niciunul din răspunsuri nu este correct

R: E

3. Se consideră algoritmul ***Contor(x, n)***, unde **n** este un număr natural ($1 \le n \le 10^3$) și **x** este un vector cu **n** elemente, numere întregi (x[1], x[2], ..., x[n]), unde $0 \le x[i] \le 10^5$, pentru i = 1, 2, ..., n, și algoritmul **Alg(numar)**, unde **numar** este un număr natural ($0 \le numar$).

```
Algorithm ALG(numar)
    If numar < 2 then Return False
    EndIf
    For d ← 2,√numar execute
        If numar MOD d = 0 then
            Return False
        EndIf
    EndForReturn True
EndAlgorithm
Algorithm CONTOR(x, n)
    count ← 0
    For i ← 1,n − 1 execute
        For j ← i + 1 to n execute
            v ← x[i]
            asc ← True
            ok ← 1
            For k ← i,j − 1 execute
                If x[k] ≥ x[k + 1] then
                    asc ← False
                    ok ← 0
                EndIf
                If ok = 1 then
                    v ← v + x[k + 1]
                EndIf
            EndFor
            If asc AND ALG(v) then
                count ← count + 1
            EndIf
        EndFor
    EndFor
    Return count
EndAlgorithm
```

Care dintre următoarele afirmații sunt adevărate?

A. Pentru x = [7, 7, 7, 7, 7, 7, 7], rezultatul returnat de Contor(x,7) este 7;

B. Pentru x = [2, 3, 5, 7], rezultatul returnat de Contor(x,4) este 2;

C. Pentru x = [7, 5, 3, 2], rezultatul returnat de Contor(x,4) este 2;

D. Pentru x = [7, 2, 2, 9, 6], rezultatul returnat de Contor(x,5) este 1.

Algoritmul returnează **numărul de subsecvențe cu mai mult de 1 element**, elementele **în ordine strict crescătoare**, a căror **sumă este un număr prim**.

R: B, D