Probleme consultații - 27 octombrie 2018

Problema 1

Enunț

Un domeniu pătrat (cu albastru), ca cel din figură trebuie apărat. Domeniul e împărțit în pătrățele. Latura domeniului este de n pătrățele, n≥7. Pe culoarele colorate patrulează 3 soldați, care încep patrularea din colțul stânga sus în același moment (poziția inițială). În exemplul de mai jos avem n=9.

Cei 3 soldați (garda) se vor schimba cu alți 3 soldați în momentul în care ajung în poziția inițială, simultan.

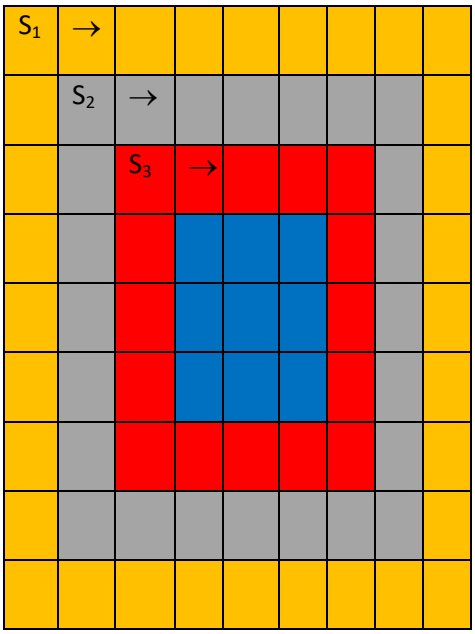
Soldații patrulează în sensul acelor de ceasornic fiecare cu aceeași viteză constantă, și fiecare pe culoarul său.

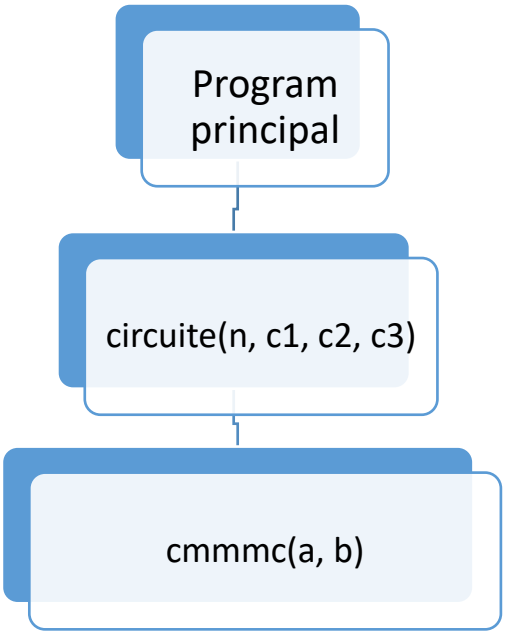
Să se implementeze un program care determină numărul de circuite complete pentru fiecare soldat (un circuit complet este făcut de un soldat oarecare, pe același culoar până ajunge în poziția inițială), pentru schimbarea gărzii.

S1 patrulează pe culoarul galben, S2 pe verde, S3 pe roșu. Albastrul este cetatea care trebuie apărată.

Analiză

- Considerăm doi soldați care patrulează circuite de lungimi diferite, măsurate în unități (o unitate este un pătrățel). Un circuit are lungime *L1*, iar al doilea lungime *L2* unități.
- Cei doi vor ajunge concomitent în punctul inițial după ce parcurg *X = cmmmc(L1, L2)* unități. Astfel, primul va parcurge *X/L1* circuite, iar al doilea *X/L2*.
- Aceeași idee se aplică pentru trei soldați.





Specificarea subalgoritmilor

Funcția **cmmmc(a, b)**:
Descriere: Calculează cel mai mic multiplu comun a două numere naturale.
Date: a, b – numere naturale.
Rezultate: cel mai mic multiplu comun al celor două numere.

Subalgoritm **circuite(n, c1, c2, c3)**:
Descriere: Calculează numarul de circuite complete pentru fiecare soldat, până la întâlnire.
Date: n – latura domeniului.
Rezultate: c1, c2, c3 – numărul de circuite parcurse de către cei trei soldați.

Implementare

Varianta C++

```
#include <iostream>

using namespace std;


/*
Descriere: Calculeaza cel mai mic multiplu comun a doua numere naturale.
Date: a, b – numere naturale.
Rezultate: cel mai mic multiplu comun al celor doua numere.*/
int cmmmc(int a, int b){
	if (a < b) {
		int aux = a;
		a = b;
		b = aux;
	}
```

```
	// cel mai mare numar se aduna repetat cu el insusi
	// si se verifica divizibilitatea cu celalalt
	int rez = a;
	while (rez % b != 0)
		rez += a;

	return rez;
}

/*
Descriere: Calculeaza numarul de circuite complete pentru fiecare soldat, pana la intalnire.
Date: n - latura domeniului.
Rezultate: c1, c2, c3 - numarul de circuite parcurse de catre cei trei soldati.*/
void circuite(int n, int& c1, int& c2, int& c3){
	int l1 = 4 * (n - 1); // lungimea culoarului exterior
	int l2 = 4 * (n - 3); // lungimea culoarului din mijloc
	int l3 = 4 * (n - 5); // lungimea culoarului interior

	int intalnire = cmmmc(cmmmc(l1, l2), l3); // cmmmc al lui l1, l2, l3
	c1 = intalnire / l1;
	c2 = intalnire / l2;
	c3 = intalnire / l3;
}

int main(){
	int n = 0;
	cout << "Introduceti latura domeniului: ";
	cin >> n;

	int c1 = 0, c2 = 0, c3 = 0;
	circuite(n, c1, c2, c3);
	cout << "Soldatul S1 va face " << c1 << " circuite pana la intalnire." << endl;
	cout << "Soldatul S2 va face " << c2 << " circuite pana la intalnire." << endl;
	cout << "Soldatul S3 va face " << c3 << " circuite pana la intalnire." << endl;

	return 0;
}
```

Varianta Pascal

```
function cmmmc(a,b:integer): integer;
                          {se aduna cel mai mare repetat cu el insusi,}
                          {si se verifica divizibilitatea cu celalalt}
var aux, rez: integer;
begin
    if(a < b)then
            begin
              aux := a;             a := b;             b := aux;
            end;

    rez := a;
    while (rez Mod b) > 0 do
        rez := rez + a;
    cmmmc := rez;
end;
```

```
procedure circuite(n:integer; var c1, c2, c3:integer);
var l1,l2,l3,intalnire: integer;     {nr. de patratele parcurse pentru un circuit
complet de cei 3 soldati}
begin
    l1:=4*(n-1);
    l2:=4*(n-3);
    l3:=4*(n-5);
    intalnire:=cmmmc(cmmmc(l1,l2),l3);   {cel mai mic multiplu comun al lui p1,p2,p3}

    c1:=intalnire div l1;              {nr de circuite pentru fiecare soldat}
    c2:=intalnire div l2;
    c3:=intalnire div l3;
end;

var n, c1, c2, c3: integer;
begin

write('Introduceti latura domeniului: ');
readln(n);
circuite(n, c1, c2, c3);
writeln('Soldatul S1 va face ', c1, ' circuite pana la intalnire.');
writeln('Soldatul S2 va face ', c2, ' circuite pana la intalnire.');
writeln('Soldatul S3 va face ', c3, ' circuite pana la intalnire.');

end.
```

Problema 2

Enunț

Să se realizeze câte o funcție recursivă cu un singur parametru (numărul *n*) pentru:

a) determinarea cifrei minime a lui *n*, se va returna cifra minimă;

b) determinarea cifrei pare maxime a lui n, se va returna cifra pară maximă sau -1, dacă *n* nu are cifre pare;

Analiză

Vom scrie formulele recursive pentru cele două cerințe.

a)

$$cifraMinima(n) = \begin{cases} n, dacă\ n\ este\ format\ dintr-o\ singură\ cifră \\ \min\{n\ mod\ 10,\quad cifraMinima\left(\left[\frac{n}{10}\right]\right), \quad altfel \end{cases}$$

O altă variantă recursivă pentru determinarea cifrei minime este:

$$cifraMinima(\overline{a_1a_2\dots a_{n-1}a_n}) = \begin{cases} n, dacă\ n\ este\ format\ dintr-o\ singură\ cifră \\ cifraMinima(\overline{a_1a_2\dots \min\{a_{n-1}, a_n\}}), \quad altfel \end{cases}$$

În această variantă apelul recursiv se face pentru numărul obținut prin înlocuirea ultimelor două cifre ale sale cu minimul dintre acestea.

b)

$$cifraParaMaxima(n) = \begin{cases} n, dacă\ e\ este\ format\ dintr-o\ singură\ cifră\ pară \\ -1, dacă\ n\ este\ format\ dintr-o\ singură\ cifră\ impară \\ cifraParaMaxima\left(\left[\frac{n}{10}\right]\right), \quad daca\ n\ mod\ 10\ este\ impara \\ \max\left\{n\ mod\ 10, cifraParaMaxima\left(\left[\frac{n}{10}\right]\right\}, daca\ n\ mod\ 10\ este\ para\right. \end{cases}$$

Specificarea funcțiilor

Funcția **cifraMinima(n)**:

Descriere: Returneaza cifra minima a unui numar dat.

Date: n – numar natural.

Rezultate: cifra minima a numarului dat.

Funcția **cifraParaMaxima(n)**:

Descriere: Returneaza cifra para maxima a unui numar dat.

Date: n – numar natural.

Rezultate: cifra para maxima a numarului dat.

Implementare

Varianta C++

```
#include <iostream>

using namespace std;

/*
Descriere:  Returneaza cifra minima a unui numar dat.
Date: n – numar natural.
Rezultate: cifra minima a numarului dat.
*/
int cifraMinima_V1(int n){
	if (n <= 9)   // cazul in care numarul este format dintr-o singura cifra
		return n;
	int cifraMinima_restul_numarului = cifraMinima_V1(n / 10);
	return (n % 10 < cifraMinima_restul_numarului ? n % 10 : cifraMinima_restul_numarului);
}

/*
Descriere:  Returneaza cifra minima a unui numar dat.
Date: n – numar natural.
Rezultate: cifra minima a numarului dat.
*/
int cifraMinima_V2(int n){
	if (n <= 9)   // cazul in care numarul este format dintr-o singura cifra
		return n;
	int minim_ultimele_doua_cifre = n % 10;
	int penultima_cifra = (n / 10) % 10;
	if (penultima_cifra < minim_ultimele_doua_cifre)
		minim_ultimele_doua_cifre = penultima_cifra;
	return cifraMinima_V2((n/100) * 10 + minim_ultimele_doua_cifre);
}

/*
Descriere:  Returneaza cifra para maxima a unui numar dat.
Date : n – numar natural.
Rezultate : cifra para maxima a numarului dat.
*/
int cifraParaMaxima(int n){
	if (n <= 9)          // daca n este format dintr-o singura cifra
	{
		if (n % 2 == 0)             // daca este par
			return n;
		else
			return -1;
	}

	int ultima_cifra = n % 10;
	if (ultima_cifra % 2 != 0)
		return cifraParaMaxima(n / 10);
	int cifraParaMaxima_restul_numarului = cifraParaMaxima(n / 10);
	return (ultima_cifra > cifraParaMaxima_restul_numarului ? ultima_cifra :
cifraParaMaxima_restul_numarului);
}
```

```
int main(){
	int n = 0;
	cout << "Introduceti numarul: ";
	cin >> n;

	cout << "Cifra minima a numarului " << n << " este: " << cifraMinima_V1(n) << endl;
	cout << "Cifra minima a numarului " << n << " este: " << cifraMinima_V2(n) << endl;
	cout << "Cifra para maxima a numarului " << n << " este: " << cifraParaMaxima(n) << endl;

	return 0;
}
```

Varianta Pascal

```
{
Descriere:  Returneaza cifra minima a unui numar dat.
Date: n - numar natural.
Rezultate: cifra minima a numarului dat.
}
function cifraMinima_V1(n: longint): integer;
var cifraMinima_restul_numarului: integer;
begin
	if (n <= 9) then // cazul in care numarul este format dintr-o singura cifra
		cifraMinima_V1 := n
    else
    begin
        cifraMinima_restul_numarului := cifraMinima_V1(n div 10);
        if ((n mod 10) < cifraMinima_restul_numarului) then
            cifraMinima_V1 := n mod 10
        else
            cifraMinima_V1 := cifraMinima_restul_numarului;
    end;
end;

function cifraMinima_V2(n: longint): integer;
var ultima_cifra, penultima_cifra: integer;
begin
    if(n > 9) then
    begin
        ultima_cifra := n mod 10;
        penultima_cifra := (n div 10) mod 10;
        if (penultima_cifra < ultima_cifra)
            then cifraMinima_V2 := cifraMinima_V2(n div 10)
            else cifraMinima_V2 := cifraMinima_V2(((n div 100) * 10) + ultima_cifra);
    end
    else cifraMinima_V2 := n;
end;
```

```
{
Descriere:  Returneaza cifra para maxima a unui numar dat.
```

```
Date : n - numar natural.
Rezultate : cifra para maxima a numarului dat.
}
function cifraParaMaxima(n: longint): integer;
var ultima_cifra, cifraParaMaxima_restul_numarului: integer;
begin
    if (n <= 9)   then  {daca n este format dintr-o singura cifra}
      begin
    if (n mod 2 = 0) then           {daca este par}
                cifraParaMaxima := n
            else
                cifraParaMaxima := -1;
      end
    else
    begin
        ultima_cifra := n mod 10;
        if (ultima_cifra mod 2 <> 0) then
            cifraParaMaxima := cifraParaMaxima(n div 10)
        else
        begin
            cifraParaMaxima_restul_numarului := cifraParaMaxima(n div 10);
            if (ultima_cifra > cifraParaMaxima_restul_numarului) then
                cifraParaMaxima := ultima_cifra
            else
                 cifraParaMaxima := cifraParaMaxima_restul_numarului;
        end;
    end;
end;

var n: longint;
begin

    write('Introduceti numarul: ');
    readln(n);
    writeln('Cifra minima este: ', cifraMinima_V1(n));
    writeln('Cifra minima este: ', cifraMinima_V2(n));
    writeln('Cifra para maxima este: ', cifraParaMaxima(n));

end.
```

Problema 3

Enunț

Fie **i,j,k** trei numere naturale. Să scrie un subprogram care returnează restul împărțirii numărului **(i^j)** la **k**, deci **(i^j)**modulo **k** (iterativ şi recursiv).

Exemple:

(100^{100}) modulo 7 = 2
(125^{199}) modulo 999 = 800
(2^{10}) modulo 9 = 7

Analiză

- practic se înmulțesc resturile modulo *k*; nu e nevoie de calculul expresiei i^j;
- se calculează la fiecare pas de înmulțire restul produsului (modulo k);

Aceasta deoarece se știe că: (a * b) mod c = (a mod c * b mod c) mod c

Atunci: a^x mod c = (a mod c * a^{x-1} mod c) mod c

Pentru varianta recursivă, formula recursivă se deduce din formula anterioară astfel:

$$rest(i,j,k) = \begin{cases} 1, & daca\ j = 0 \\ ((i\ mod\ k) * rest(i, j-1, k))\ mod\ k \end{cases}$$

Specificarea funcției

Funcția **Rest(i, j, k)**:
Descriere: returnează restul împărțirii lui (i^j) la k.
Date: i,j,k - numere naturale
Rezultate: R un număr natural: R∈{0,1,...k-1}

Implementare

Varianta iterativă C++/*

```
Descriere:  returneaza restul impartirii lui (i^j) la k.
Date: i,j,k - numere naturale
Rezultate: R un numar natural: R in {0,1,...k-1}.
*/
int Rest(int i, int j, int k){
	int r = i % k;
	int rest = 1;
	while (j > 0) {
		rest = (rest * r) % k;
		j--;
	}
	return rest;
}
```

Varianta recursivă C ++

```
/*
Descriere:  returneaza restul impartirii lui (i^j) la k.
Date: i,j,k - numere naturale
Rezultate: R un numar natural: R in {0,1,...k-1}.
*/
int Rest(int i, int j, int k)
{
	if (j == 0)
		return 1;
	return ((i % k) * Rest(i, j - 1, k)) % k;
}
```

Varianta iterativă Pascal

```
function Rest(i,j,k:integer):integer;      {functie ce determina restul}
var R,iModK:integer;
begin                                      {e suficient sa inmultim resturile}
   iModK:=i mod k;                         {expresia i Mod k e constanta in ciclu}
   R    :=1;
   While(j>0) do
      begin
        R:=(R*(iModK)) mod k;        {atat lui i cat si produsului se aplica mod}
        j:=j-1;
      end;
   Rest:=R;
end;
```

Varianta recursivă Pascal

```
function Rest (i,j,k:integer):integer);
begin
    if(j>0) then Rest:= ((i mod k)*Rest(i,j-1,k)) % k
            else Rest:=1;
end;
```

Exemple

Date de intrare	Rezultate
100, 100, 7	2
125, 199, 999	800
2, 10, 9	7
8, 0, 100	1
5, 151, 5	0

Problema 4

Enunț

Fie **X,Y** două numere naturale. Să scrie un subprogram care determină dacă două numere date sunt **asemenea**, fără a folosi tablouri. (Două numere naturale sunt **asemenea** dacă au aceleaşi cifre: **2131** e **asemenea** cu **32211** pentru că mulțimea cifrelor este aceeași: {**1,2,3**}).

Analiză

Problema are o rezolvare simplă dacă se folosește conceptul de vector de apariție pentru cifrele unui număr:

- se determină vectorii de apariție pentru **X** şi **Y**;
- se compară apoi vectorii de apariție și se decide asemănarea.

Fără utilizarea vectorilor ar trebui să vedem dacă fiecare cifra a lui **X** este în **Y** şi invers.

Vom face 2 subprograme:

- Un subprogram care verifică dacă fiecare cifră a unui număr se află printre cifrele altui număr.
- Un subprogram care apelează apoi de 2 ori primul subprogram.

Specificarea subalgoritmilor

Functia **CifreAinB(A, B)**:

Descriere: verifică dacă fiecare cifră a lui A este în mulțimea cifrelor lui B.

Date: **A,B > 0** numere naturale

Rezultate: true, dacă mulțimea cifrelor lui **A** este inclusă în mulțimea cifrelor lui **B;**
false, în caz contrar

Functia **Asemenea(X, Y)**:

Descriere: verifică dubla incluziune a cifrelor lui X şi Y.

Date: - **X,Y > 0** numere naturale

Rezultate: true, dacă **CifreAinB(X,Y)** este true **ŞI** **CifreXinY(Y,X)** este true;
false, în caz contrar

Implementare

Varianta C++

```
/*
Descriere: verifica daca fiecare cifra a lui A este in multimea cifrelor lui B.
Date: A, B > 0 numere naturale
Rezultate:    true, daca multimea cifrelor lui A este inclusa in multimea cifrelor lui B;
                    false, in caz contrar*/
bool CifreAinB(long a, long b){
       int copieB = b;      // se retine o copie a lui B
       while (a > 0){       // vom verifica daca fiecare cifra a lui A este in B
              int ultima_cifra_A = a % 10;
              while (copieB > 0 && ultima_cifra_A != copieB % 10)
                    copieB /= 10;
              if (copieB == 0)// daca s-a ajuns la 0, cifra curenta a lui A nu este in B
                    return false;
              copieB = b;                 // se reinitializeaza copia cu numarul initial B
```

```
            a /= 10;                    // se trece la urmatoarea cifra a lui A
        }
        return true;
}

/*
Descriere:  verifica dubla incluziune a cifrelor lui x si y.
Date: x, y > 0 numere naturale
Rezultate:    true, daca CifreAinB(x, y) este true SI CifreAinB(y,x) este true;
                    false, in caz contrar*/
bool Asemenea(long x, long y){
        return (CifreAinB(x, y) && CifreAinB(y, x));
}
```

Varianta Pascal

```
function CifreAinB(A,B:longint):boolean;
  var CopieB  :longint;
      UcifA   :byte;
  begin
    CopieB   :=B;                   {se retine o clona a lui B in CopieB}
    CifreAinB:=true;                {presupunem ca incluzinea exista}
    while (A>0) do
      begin
        UcifA:=A mod 10;            {verifica fiecare cifra a lui A daca e B}
        B    :=CopieB;              {la fiecare cifra a lui A se reface B}
        while (B>0) and (UcifA<>(B mod 10)) do B:=B div 10;
        if (B=0)                    {daca B=0 => cifra curenta lui A nu e in B}
          then
            begin
              CifreAinB:=false;     {se pune numele functiei pe false si A pe 0}
              A        :=0;         {pentru a iesi din primul ciclu while}
            end
          else A:=A div 10;         {daca B>0, cifra curenta a lui A e in B}
      end;                          {se trece la următoarea cifra a lui A}
  end;


 function Asemenea(X,Y:longint):boolean;
 begin
      if (CifreAinB(X,Y)) and (CifreAinB(Y,X))
            then Asemenea:=true
            else Asemenea:=false;
 end;
```

Exemple

Date de intrare	Rezultate
1222331, 123	true
122235, 123	false
5656565, 56	true
5656565, 5	false
1, 8	false

Problema 5

Enunț

La un concurs participă **n** elevi. Ei se împart în **m** echipe astfel încât fiecare echipă să aibă cel puțin un participant.
În timpul concursului se creează prietenii între fiecare pereche de participanți ai aceleași echipe. Dându-se **n** și **m** să se afișeze numărul minim și maxim de perechi care se pot forma.

Exemple:

a) n=5, m=1 => minim = maxim = 10
b) n=3, m=2 => minim = maxim = 1
c) n=6, m=3 => minim = 3, maxim = 6
d) n=10, m=3 => minim = 12, maxim = 28.

Analiză

Numărul de perechi dintr-o echipă este $C_k^2 = \frac{k(k-1)}{2}$, unde *k* este numărul membrilor echipei.
Echipele de câte un membru nu contribuie cu nimic la numărul total de perechi.

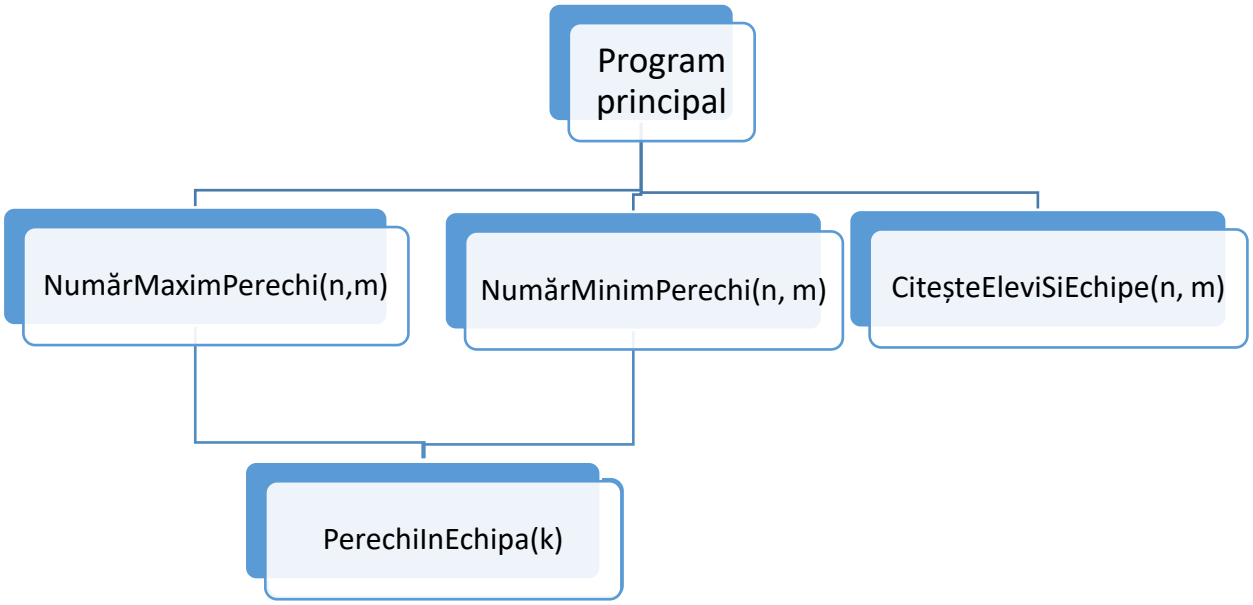
Exemplu:

```
n = 14
m = 3

1, 1, 12 = (12 * 11) / 2 = 66
1, 2, 11 = 1 + (11* 10) / 2 = 56
1, 3, 10 = 3 + (10 * 9) / 2 = 48
1, 4, 9 = (4 * 3) / 2 + (9 * 8) / 2 = 6 + 36 = 42
1, 5, 8 = (5 * 4) / 2 + (8 * 7) / 2 = 10 + 28 = 38
1, 6, 7 = (6 * 5) / 2 + (7 * 6) / 2 = 15 + 21 = 36
2, 2, 10 = 1 + 1 + (10 * 9) / 2 = 47
2, 3, 9 = 1 + 3 + (9 * 8) / 2 = 40
2, 4, 8 = 1 + (4 * 3) / 2 + (8 * 7) / 2 = 1 + 6 + 28 = 35
2, 5, 7 = 1 + (5 * 4) / 2 + (7 * 6) / 2 = 1 + 10 + 21 = 32
2, 6, 6 = 1 + (6 * 5) = 31
3, 3, 8 = 3 + 3 + (8 * 7) / 2 = 6 + 28 = 34
3, 4, 7 = 3 + (4 * 3) / 2 + (7 * 6) / 2 = 3 + 6 + 21 = 30
3, 5, 6 = 3 + (5 * 4) / 2 + (6 * 5) / 2 = 3 + 10 + 15 = 28
4, 4, 6 = (4 * 3) + (6 * 5) / 2 = 12 + 15 = 27
4, 5, 5 = (4 * 3) / 2 + (5 * 4) = 26
```

- Pentru a se forma numărul maxim de perechi, avem nevoie de o echipă cât mai mare și restul de câte 1 membru. Deci vor fi (m-1) echipe de câte un membru și o echipă de (n-m+1) membri.

- Pentru a se forma numărul minim avem nevoie de echipe al căror număr de membri să nu difere cu mai mult de 1. Vom avea astfel [n/m] echipe, iar restul n % m de membri se vor imparti cate unul la fiecare echipa deja formată.



Specificarea subalgoritmilor

Functia **`PerechiInEchipa(k)`**:
Descriere: Determina cate perechi se pot forma intr-o echipa de k membri.
Date: k - numar natural
Rezultate: numarul de perechi care se pot forma intr-o echipa cu k membri.

Functia **`NumarMaximPerechi(n, m)`**:
Descriere: Determina numarul maxim de perechi care se pot forma, considerand n membri si m echipe.
Date: n, m - numere naturale
Rezultate: numarul maxim de perechi care se pot forma, considerand n membri si m echipe.

Functia **`NumarMinimPerechi(n, m)`**:
Descriere: Determina numarul minim de perechi care se pot forma, considerand n membri si m echipe.
Date: n, m - numere naturale
Rezultate: numarul minim de perechi care se pot forma, considerand n membri si m echipe.

Subalgoritm **`CitesteEleviSiEchipe(n, m)`**:
Descriere: Citeste numarul de elevi si numarul de echipe.
Date: -
Rezultate: n, m - numere naturale, reprezentand numarul de elevi si numarul de echipe.

Implementare

Varianta C++

```cpp
#include <iostream>

using namespace std;

/*
Descriere: Determina cate perechi se pot forma intr-o echipa de k membri.
Date: k - numar natural
Rezultate: numarul de perechi care se pot forma intr-o echipa cu k membri.*/
int PerechiInEchipa(int k){
	return k * (k - 1) / 2;
}

/*
Descriere: Determina numarul maxim de perechi care se pot forma, considerand n membri si m echipe.
Date: n, m - numere naturale
Rezultate: numarul maxim de perechi care se pot forma, considerand n membri si m echipe.*/
int NumarMaximPerechi(int n, int m){
	return PerechiInEchipa(n - m + 1);
}

/*
Descriere: Determina numarul minim de perechi care se pot forma, considerand n membri si m echipe.
Date: n, m - numere naturale
Rezultate: numarul minim de perechi care se pot forma, considerand n membri si m echipe.*/
int NumarMinimPerechi(int n, int m){
	int nrMinimMembriIntr_oEchipa = n / m;
	int nrEchipeCuMaiMulti = n % m;
	return PerechiInEchipa(nrMinimMembriIntr_oEchipa) * (m - nrEchipeCuMaiMulti) +
PerechiInEchipa(nrMinimMembriIntr_oEchipa + 1) * nrEchipeCuMaiMulti;
}

/*
Descriere: Citeste numarul de elevi si numarul de echipe.
Date: -
Rezultate: n, m - numere naturale, reprezentand numarul de elevi si numarul de echipe.*/
void CitesteEleviSiEchipe(int& n, int& m){
	cout << "Dati numarul elevilor: ";
	cin >> n;
	cout << "Dati numarul echipelor: ";
	cin >> m;
}

int main(){
	int n = 0, m = 0;
	CitesteEleviSiEchipe(n, m);

	cout << "Numarul maxim de perechi este: " << NumarMaximPerechi(n, m) << endl;
	cout << "Numarul minim de perechi este: " << NumarMinimPerechi(n, m) << endl;

	return 0;
}
```

Varianta 2 - în cazul în care se cere un singur subalgoritm care primeste **n** și **m** calculează numărul minim și maxim de perechi.

```
void NumarMinimSiMaximDePerechi(int n, int m, int& nrMinim, int& nrMaxim){
	int rest = n % m;
	int cat = n / m;
	nrMinim = m * cat * (cat - 1) / 2 + cat * rest;
	nrMaxim = (n - m + 1)*(n - m) / 2;
}
```

Varianta Pascal

```
{
Descriere: Determina cate perechi se pot forma intr-o echipa de k membri.
Date: k - numar natural
Rezultate: numarul de perechi care se pot forma intr-o echipa cu k membri.
}
function PerechiInEchipa(k: integer): integer;
begin
	PerechiInEchipa := k * (k - 1) div 2;
end;

{
Descriere: Determina numarul maxim de perechi care se pot forma, considerand n membri si
m echipe.
Date: n, m - numere naturale
Rezultate: numarul maxim de perechi care se pot forma, considerand n membri si m echipe.
}
function NumarMaximPerechi(n, m: integer): integer;
begin
	NumarMaximPerechi := PerechiInEchipa(n - m + 1);
end;

{
Descriere: Determina numarul minim de perechi care se pot forma, considerand n membri si
m echipe.
Date: n, m - numere naturale
Rezultate: numarul minim de perechi care se pot forma, considerand n membri si m echipe.
}
function NumarMinimPerechi(n, m: integer): integer;
var nrMinimMembriIntr_oEchipa, nrEchipeCuMaiMulti: integer;
begin
	nrMinimMembriIntr_oEchipa := n div m;
	nrEchipeCuMaiMulti := n mod m;
	NumarMinimPerechi := PerechiInEchipa(nrMinimMembriIntr_oEchipa) * (m -
			nrEchipeCuMaiMulti) +
			PerechiInEchipa(nrMinimMembriIntr_oEchipa + 1) * nrEchipeCuMaiMulti;
end;

{
Descriere: Citeste numarul de elevi si numarul de echipe.
Date: -
Rezultate: n, m - numere naturale, reprezentand numarul de elevi si numarul de echipe.
}
procedure CitesteEleviSiEchipe(var n: integer; var m: integer);
```

```
begin
      write('Dati numarul elevilor: ');
      readln(n);
      write('Dati numarul echipelor: ');
      readln(m);
end;

var n, m: integer;
begin
    n := 0;
    m := 0;
      CitesteEleviSiEchipe(n, m);

      writeln('Numarul maxim de perechi este: ', NumarMaximPerechi(n, m));
      writeln('Numarul minim de perechi este: ', NumarMinimPerechi(n, m));
end.
```

Varianta 2 - în cazul în care se cere un singur subalgoritm care primeste **n** și **m** calculează numărul minim și maxim de perechi.

```
procedure NumarMinimSiMaximDePerechi(n, m: integer; var nrMinim: integer; var nrMaxim:
integer);
var rest, cat: integer;
begin
      rest := n mod m;
      cat := n div m;
      nrMinim := m * cat * (cat - 1) div 2 + cat * rest;
      nrMaxim := (n - m + 1)*(n - m) div 2;
end;
```

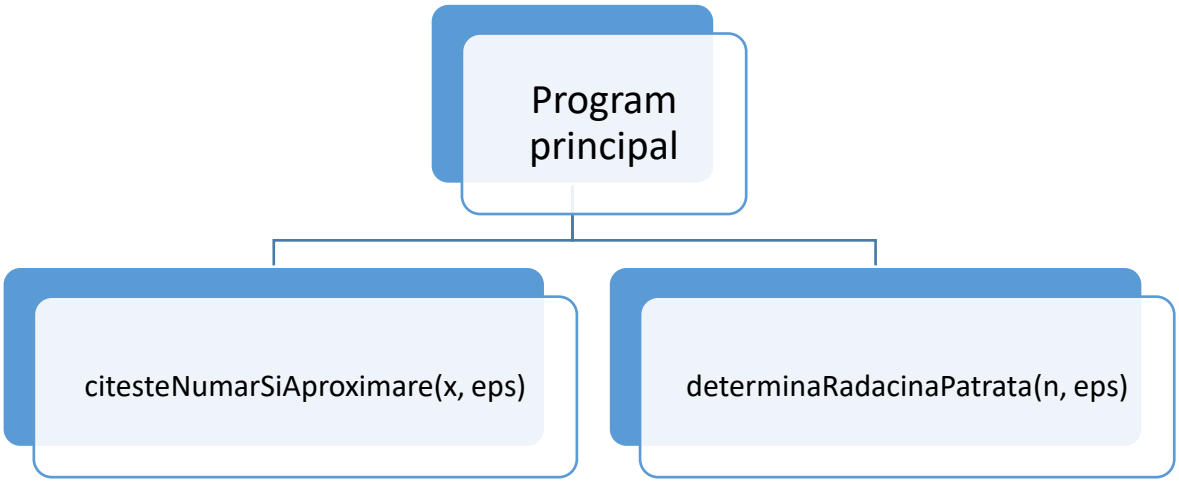
Problema 6

Enunț

Dat fiind un număr real *x*, să se determine rădăcina pătrată *r* a acestuia, cu aproximarea *epsilon*.

Analiză

Se va folosi metoda înjumătățirii intervalului: pornind de la un interval inițial pentru rădăcina pătrată, se selectează iterativ subintervalul în care se află rădăcina, până când se ajunge la o lungime maximă a intervalului mai mică decât aproximarea cerută.



Specificarea subalgoritmilor

- Subalgoritmul **citesteNumarSiAproximare(x, eps)**:
  Descriere: Citește două numere reale.
  Date: -
  Rezultate: *x, eps* – numere reale, x > 0.

- Subalgoritmul **determinaRadacinaPatrata(n, eps)**:
  Descriere: Determina rădăcina pătrata a unui număr dat, folosind o aproximare dată.
  Date: x, eps – numere reale, x > 0
  Rezultate: rădăcina pătrată a lui *x*, cu aproximarea *eps*.

Implementare

Varianta C++

```
/*
Descriere: Citeste doua numere reale, reprezentand numarul caruia trebuie sa ii fie calculata
radacina patrata si aproximarea (eroarea).
Date: -
Rezultate: se citesc cele doua numere.*/
void citesteNumarSiAproximare(double& x, double& eps){
	cout << "Introduceti numarul real si apoi precizia: ";
	cin >> x >> eps;
}

/*	Descriere: Determina radacina patrata a unui numar dat, folosind o aproximare data.
	Date: x, eps - numere reale.
	Rezultate: radacina patrata a lui x, cu aproximarea eps.*/
double determinaRadacinaPatrataIterativ(double x, double eps){
	double dreapta = x;
	double stanga = 0;
	double mij = (dreapta + stanga) / 2;
	while (abs(dreapta - stanga) > eps)
	{
		if (mij * mij > x)
			dreapta = mij;
		else
			stanga = mij;
		mij = (dreapta + stanga) / 2;
	}
	return mij;
}

/*	Descriere: Determina radacina patrata a unui numar dat, folosind o aproximare data.
	Date: x, eps - numere reale.
	Rezultate: radacina patrata a lui x, cu aproximarea eps.*/
double determinaRadacinaPatrataRecursiv(double x, double eps){
	return determinaRadacinaPatrataRec(x, eps, 0, x);
}

double determinaRadacinaPatrataRec(double x, double eps, double stanga, double dreapta){
	double mij = (dreapta + stanga) / 2;
	if (abs(dreapta - stanga) < eps)
		return mij;
	if (mij * mij > x)
		return determinaRadacinaPatrataRec(x, eps, stanga, mij);
	return determinaRadacinaPatrataRec(x, eps, mij, dreapta);
}

int main(){
	double x = 0, eps = 0;
	citesteNumarSiAproximare(x, eps);
	cout << "Iterativ: ";
	cout << "Radacina patrata a numarului " << x << ", cu aproximarea " << eps << " este: " <<
determinaRadacinaPatrataIterativ(x, eps) << endl;
	cout << endl;
	cout << "Recursiv: ";
```

```
	cout << "Radacina patrata a numarului " << x << ", cu aproximarea " << eps << " este: " <<
determinaRadacinaPatrataRecursiv(x, eps) << endl;
	return 0;
}
```

Varianta Pascal

```
procedure citesteNumarSiAproximare(var x: real; var eps: real);
begin
    writeln('Introduceti numarul real si apoi aproximarea: ');
    readln(x, eps);
end;

function determinaRadacinaPatrataIterativ(x, eps: real): real;
var dreapta, stanga, mij: real;
begin
    dreapta := x;
    stanga := 0;
    mij := (dreapta + stanga) / 2;

    while (abs(dreapta - stanga) > eps) do
      begin
            if (mij * mij > x) then
                  dreapta := mij
            else
                  stanga := mij;

            mij := (dreapta + stanga) / 2;
      end;

    determinaRadacinaPatrataIterativ := mij;
end;

function determinaRadacinaPatrataRec(x, eps, stanga, dreapta: real): real;
var mij: real;
begin
      mij := (dreapta + stanga) / 2;

      if (abs(dreapta - stanga) < eps) then
            determinaRadacinaPatrataRec := mij
      else
      if (mij * mij > x) then
            determinaRadacinaPatrataRec := determinaRadacinaPatrataRec(x, eps, stanga,
mij)
          else
              determinaRadacinaPatrataRec := determinaRadacinaPatrataRec(x, eps, mij,
dreapta);
end;
```

```
function determinaRadacinaPatrataRecursiv(x, eps: real): real;
begin
      determinaRadacinaPatrataRecursiv := determinaRadacinaPatrataRec(x, eps, 0, x);
end;

var x, eps: real;
begin
    citesteNumarSiAproximare(x, eps);
    writeln('Iterativ: ');
    writeln('Radacina patrata cu aproximarea data este: ',
determinaRadacinaPatrataIterativ(x, eps));
    writeln('Recursiv: ');
    writeln('Radacina patrata cu aproximarea data este: ',
determinaRadacinaPatrataRecursiv(x, eps));
end.
```

Exemple

Date de intrare	Rezultate
5, 0.01	2.23145
5, 0.0001	2.23606
100, 0.0000001	10
100, 0.1	10.0098
2, 0.03	1.41406

Probleme tip grilă

1. Știind că variabilele *a* și *i* sunt întregi, stabiliți ce reprezintă valorile afișate de algoritmul de mai jos. S-au folosit notațiile *x%y* pentru restul împărțirii numărului întreg *x* la numărul întreg *y*, și *[x]* pentru partea întreagă a numărului real *x*.

```
a ← 10
pentru i ← 1,6 execută
      scrie [a/7]
      a ← a % 7 * 10
sfârșit pentru
```

a. primele 6 zecimale ale lui 1/7
b. primele 7 zecimale ale lui 1/6
c. primele 6 cifre ale lui 10/7
d. primele 7 cifre ale lui 10/6

2. Ce va afișa algoritmul pseudocod de mai jos pentru două numere naturale nenule *a* și *b*? S-a notat cu *x%y* restul împărțirii numerelor întregi *x* și *y*.

```
citește a,b
c ← 1
cât-timp a * c % b ≠ 0 execută
     c ← c + 1
sfarsit-cat-timp
scrie a*c
```

a. a^b
b. cel mai mic multiplu comun
c. cel mai mare divizor comun
d. a*b

3. Care este rezultatul execuției următoarei secvențe?
   a. Afişează 3 pentru n=123 şi c=3;
   b. Afişează 2 pentru n=12003 şi c=0;
   c. Afişează n, pentru n=c=1;
   d. Afişează 4, pentru n=1277771 şi c=7
   e. Afişează 3, pentru n=12555 şi c=5.

```
citeşte n,c {n, natural>0, c cifra}
z←0
┌cât timp n%10=c execută
│ n←[n/10]
│ z←z+1
└■
scrie z
```

4. Care este rezultatul execuției următoarei secvențe?
   a. Afişează x, dacă m<=0.
   b. Afişează x^2, dacă m=2.
   c. Afişează 1, pentru m <0.
   d. Afişează x^k, dacă $m=2^k$.
   e. Afişează un număr negativ dacă x<0.

```
Citeşte x,m {x,m întregi}
y←1
┌Cât timp m>0 execută
│  Dacă m%2=0
│  │  atunci m← [m/2]; x←x*x
│  │  altfel   m←m-1; y←y*x
│  └■
└■
Afişează y
```

Care este rezultatul următoarei secvențe?

a. suma numerelor divizibile cu 2, mai mici ca n.
b. suma numerelor divizibile cu 2 şi 4 şi mai mici ca n.
c. suma numerelor divizile cu puteri ale lui 2.
d. exponentul lui 2 (ca factor) în n!
e. exponentul lui 4 (ca factor) în n!

```
Citeste n
  p←2
  S←0
 ┌CâtTimp (p<=n)
 │  S←s+[n/p];
 │  p←p*2;
 └■
Afişează s
```

5. Care funcţie schimbă valorile întregi ale lui a şi b între ele?

```
A.
void F(int& a, int& b){
      a-=b;
      b+=a;
      a=b-a;
}
```

```
B.
void F(int& a, int& b){
      a=a+b;
      b=a-b;
      a=a-b;
}
```

```
C.
void F(int& a, int& b){
      a=a/b;
      b=a*b;
      a=b/a;
}
```

```
D.
void F(int& a, int& b){
      a=a*b;
      b=a/b;
      a=a/b;
}
```

```
E.
void F(int& a, int& b){
      b=b-a;
      b=a*b;
      a=b;
}
```

6. Care subprogram determină dacă numărul întreg n este prim (*true* dacă n e prim, *false* altfel)?

```
a)
bool Prim(int n){
  int d=2;
  while(d*d<=n){
     if(n%d==0) return false;
     d=(d==2)?3:d+2;
  }
  return true;
}
```

```
c)
bool Prim(int n, int d){//apel
extern, d=2
  if(n<=1) return false;
  if(d*d>n) return true;
  if(n%d==0) return false;
  return Prim(n,(d==2)?3:d+2);
}
```

7. Se consideră funcția de mai jos, care primește ca parametru un număr natural.

```
int f(int a)
{
     int x = a % 10;

     if (x == a)
            if (x % 2 == 0)
                   return a;
            else
                   return 0;

    if (x % 2 == 0)
           return 10 * f((int)(a/10)) + x;
    return f((int)(a/10));
}
```

I. De câte ori se apelează funcția pentru a = 253401976?
   a. De 3 ori
   b. De 9 ori
   c. De 6 ori
   d. De 7 ori

II. Care este rezultatul funcției pentru a = 253401976?
   a. 206
   b. 53019
   c. 2406
   d. 0