1. Se consideră expresia logică: **(X OR Z) AND (NOT X OR Y).** Care din variantele de mai jos fac expresia true (adevărată)?
   a. X ← false; Y ←false; Z ← true;
   b. X ← true; Y ←false; Z ← false;
   c. X ← false; Y ←true; Z ← false;
   d. X ← true; Y ←true; Z ← true;

2.

```
n=100;
i=0;
while(i<n)
    i++;
cout<<i;
```

Ce valoare este afişată?
   a) a lui n;
   b) 99;
   c) 101;
   d) 100;

3. Care funcţie schimbă valorile întregi, nenule 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;
}
```

4. Ce returnează funcţia F pentru n, număr întreg?

```
int F(int n) {
  int u,p;
  if(n<10) return n;
  u=n%10;
  p=F(n/10);
  if (u>p) return u;
           return p;
}
```

a) 0, pentru n=0;
b) 9, pentru n=798659;
c) 5, pentru n=598765;
d) Cifra 9, pentru n>0;

5. Ce valori vor avea a, b şi k la sfârşitul secvenţei?

```
int a=20, b=70, k=0;
if(a<b) {
  b*=a; a=b/a; b/=a;
}
while(a>=b){
   a-=b;
   k+=2;
}
```

a) 70,70,5;
b) 70,20,2;
c) 10,20,6;
d) 10,30,4;

6. Ce face subprogramul pentru n întreg?

```
void sp(long n){
  for(int i=2;i<=n;i++)
   {int c=0;
    while(!(n%i)){
       n/=i;
       c+=1;
    }
    if(c) cout<<i<<" "<<c<<endl;
   }
}
```

a) Afişează perechi de numere prime (n≥2);
b) Afişează doar factorii primi ai lui n (n≥2);
c) Afişează factorii primi ai lui n şi puterile lor (n≥2);
d) Subprogramul nu afişează nimic, dacă n<2;

7. Ce face secvenţa de instrucţiuni?

```
citeşte a,b {numere întregi}
x ←1
┌cât timp (a>0)şi(b>0) execută
│ ┌dacă (a mod 10)<(b mod 10)
│ │  atunci x←0
│ └■
│  a←[a/10]
│  b←[b/10]
└■
┌dacă (x=1) şi (b=0)
│ atunci scrie "DA"
│ altfel scrie "NU"
└■
```

a) Afişează „NU” pentru perechi de numere strict negative, a,b<0;
b) Afişează „DA” dacă a<b;
c) Afisează „DA” dacă fiecare cifră a lui a este ≥decât cifra corespondentă a lui b (unităţi, zeci, etc.) şi nrCif(b)< nrCif(a), a,b>0;
d) Afişează „DA” pentru a=34, b=12;

8. Ce face secvenţa de instrucţiuni?

```
long n,B;
cin>>n;
B = n%10;
while (n>9)
  n=n/10;
if(B%2 == n%2)       cout<<"DA";
else                 cout<<"NU";
```

a) Afişează „DA” pentru n>0;
b) Afişează „DA” dacă ultima cifră a lui n este egală cu penultima cifră a lui n (n>0 şi n<100);
c) Afişează „NU” dacă prima cifră a lui n are aceeaşi paritate cu ultima cifră a lui n, n>0;
d) Afişează „DA” pentru **n=-1233**;

9. Ce face secvenţa de instrucţiuni pentru şirul: 5, 24, 5, 25, 5, 26, 0.

```
citeşte x {x natural}
nr←0
s←0
┌cât timp x≠0 execută
│ nr←nr+1
│ ┌dacă nr%2=0 atunci
│ │ s←s+x%10
│ └■
│ citeşte x
└■
scrie s,nr
```

a) Afişează 15 6;
b) Afişează 10 9;
c) Afişează 15 3;
d) Afişează doar un număr (=suma ultimelor cifre ale numerelor indice par din şirul citit);

10. Ce face secvenţa?

```
int i,n;  //n>0
cin>>n;
i=1;
while (n){
  if(n%2) cout<<i<<" ";
  i=i+1;
  n=n/2;
}
```

a) Afişează secvenţa: 1 2 3 4 5 pentru n=31;
b) Afişează secvenţa: 2 3 4 pentru n=14;
c) Afişează 1 la începutul secvenţei, pentru n impar,n>0;
d) Afişează un singur număr pentru $n=2^k$

11. Ce se afișează?. Se consideră următorul program:

Varianta C

```
#include <stdio.h>
int prelVector(int v[], int *n) {
    int s = 0; int i = 2;
    while (i <= *n) {
            s = s + v[i] - v[i - 1];
            if (v[i] == v[i - 1])
               *n = *n - 1;
            i++;
     }
  return s;
}

int main(){
    int v[8];
    v[1] = 1; v[2] = 4; v[3] = 2;
    v[4] = 3; v[5] = 3; v[6] = 10;
    v[7] = 12;
    int n = 7;
    int rezultat = prelVector(v, &n);
    printf ("%d;%d", n, rezultat);
    return 0;
}
```

Varianta C++

```
#include <iostream>
using namespace std;
int prelVector(int v[], int&n) {
    int s = 0; int i = 2;
    while (i <= n) {
            s = s + v[i] - v[i - 1];
            if (v[i] == v[i - 1])
                    n--;
            i++;
    }
return s;
}

int main(){
    int v[8];
    v[1] = 1; v[2] = 4; v[3] = 2;
    v[4] = 3; v[5] = 3; v[6] = 10;
    v[7] = 12;
    int n = 7;
    int rezultat = prelVector(v, n);
    cout << n <<";" << rezultat;
    return 0;
}
```

Varianta PasCal

```
type vector=array [1..10] of integer;
function prelVector(v: vector;
var n: integer): integer;
var s, i: integer;
begin
    s := 0; i := 2;
    while (i <= n) do
        begin
            s := s + v[i] - v[i - 1];
            if (v[i] = v[i - 1]) then n := n - 1;
            i := i + 1;
        end;
    prelVector := s;
end;

var n, rezultat:integer; v:vector;
begin
            n := 7;
            v[1] := 1; v[2] := 4; v[3] := 2;
            v[4] := 3; v[5] := 3; v[6] := 10;
            v[7] := 12;
            rezultat := prelVector(v,n);
            write(n, ';', rezultat);
end.
```

Precizați care este rezultatul afișat în urma executării programului.

A. 7;11 B. 6;9 C. 7;9 D. 7;12

12. Ce returnează funcţia F pentru n, număr întreg?

```
int F(int n) {
  int u,p;
  if(n<10) return n;
  u=n%10;
  p=F(n/10);
  if (u<p) return u;
           return p;
}
```

a) 0, pentru n ≤0;
b) 9, pentru n=98765;
c) 5, pentru n=98567.
d) 0, pentru n>0;

13. Ce afişează secvenţa?

```
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
```

a) Afişează x^2, dacă m=2;
b) Afişează 1, pentru m <0;
c) Afişează x^k, dacă m=2^k ;
d) Afişează un număr negativ dacă x<0;

14. Ce afişează secvenţa?

```
Citeşte n {număr natural ≥ 0 }
t←1; c←n%10; n←[n/10]
┌Cât timp t=1 şi n>0 execută
│  Dacă n%10>c
│  │   atunci t←0
│  └■
│  c←n%10; n←[n/10]
└■
Afişează t
```

a) Afişează 1, dacă n>0;
b) Afişează 0, dacă n=1234;
c) Afişează 0, dacă n=4321;
d) Afişează 1, dacă n=55555;

15. Care subprogram determină primalitatea lui **n**, întreg: **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;
}
```

```
b)
bool Prim(int n){
  if(n<=1) return false;
  for (int d=2; d*d<=n; d=(d==2)?3:d+2)
     if(n%d==0) return false;
  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);
}
```

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

16. Ce face secvenţa (n este natural, n>0)

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

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

17. Care din următoarele propoziţii sunt adevărate?

```
citeşte n (număr natural cu cel mult 9 cifre)
┌cât timp n≥10
│ s←0
│┌cât timp n≠0 execută
│└  s←s+n%10
│   n← [n/10]
└■
  n←s
■
scrie n
```

a) corpul ciclului exterior se execută de n ori;
b) indiferent de valoarea lui n, se afişează cifra 9;
c) indiferent de valoarea lui n, se afişează o cifră;
d) corpul ciclului exterior se execută de cel mult 3 ori;

18. Ce se afişează?

```
citeşte n  (nr. natural,n>1)
 d←2 (d număr natural)
 ┌ cat timp n%d≠0 execută
 │    d←d+1
 └■
 ┌ cat timp n%d=0 execută
 │    n← [n/d]
 └■
 ┌ dacă n=1 atunci scrie d
 │          altfel scrie n
 └■
```

a) val. iniţială a lui n, dacă n este prim;
b) 1, dacă n este prim;
c) 2, dacă n este $n=2^k$;
d) val. iniţială a lui n, dacă val. iniţială a lui n nu este număr prim;

19. Ce se afişează (n întreg)?

```
citeşte n
i←2; p←1
┌cât timp n>1 execută
│ k←0
│ ┌cât timp i|n execută  //n e divizibil cu i
│ │ k←i; n←[n/i]
│ └■
│ ┌dacă k≠0 atunci
│ │ p←p*k
│ └■
│ i←i+1
└■
scrie p
```

a) **1**, pentru **n=-6**;
b) **6**, dacă **n=6**;
c) **n!**, dacă **n>0**;
d) produsul factorilor primi ai lui n, n natural;

20. Care e valoarea lui z?

```
citeşte n {n număr natural}
z←0
┌cât timp n>0 execută
│ c←n%10; n←[n/10]
│ ┌dacă c%2=0
│ │ atunci z←z*10+c
│ └■
└■
scrie z
```

a) Dacă n=12345, z este 12345;
b) Dacă n=54321, z este 531;
c) Daca n=135, z=0;
d) Dacă n=12345, z=42;

21. **Expresie logică.**
Precizați care dintre următoarele expresii are valoarea adevărat dacă și numai dacă numărul natural n este divizibil cu 3 și are ultima cifră 4 sau 6:

A. n DIV 3 = 0 și (n MOD 10 = 4 sau n MOD 10 = 6)
B. n MOD 3 = 0 și (n MOD 10 = 4 sau n MOD 10 = 6)
C. (n MOD 3 = 0 și n MOD 10 = 4) sau (n MOD 3 = 0 și n MOD 10 = 6)
D. (n MOD 3 = 0 și n MOD 10 = 4) sau n MOD 10 = 6

22. Ce face?

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

a) Afişează 3 pentru n=123 şi c=3;
b) Afişează 2 pentru n=12003 şi c=0;
c) Afişează 4, pentru n=1277771 şi c=7;
d) Afişează 3, pentru n=12555 şi c=5;

23. Ce face?

```
citeşte n,k {n,k naturale>0}
z←0
┌pentru i=1,n execută
│   ┌dacă k|i atunci z←z+i //sau i¦k
│   └■
└■
scrie z
```

a) Afişează **1** pentru **n>k**;
b) Afişează $k * \left(1 + \left[\frac{n}{k}\right]\right) * \left[\frac{n}{k}\right] / 2$ pentru **n>k**;
c) Afişează **105** pentru **n=40** şi **k=7**;
d) Afişează o putere a lui **2**, pentru **n>k=2**;

24. Ce returnează funcţia F pentru n, număr natural?

```
long F(long n){
   long u,p;
   if(n<10)  return (n%2==0)? n:-1;
   u=(n%2==0)?(n%10):-1;
   p=F(n/10);
   return (u>p)?(u):p;
}
```

a) **n**, pentru **n=0**;
b) **6**, pentru n=**7625**;
c) **5**, pentru n=**98765**;
d) Cifra **8**, pentru **n>0**;

25. Ce valoare se afişează?

```
int n=50,i;
for(i=3;i<n;i+=5){
      //corp
      //nu se modifică i
   }
cout<<i;
```

a) **n-1**;
b) **50**;
c) **53**;
d) **48**;

26. Ce valoare se returnează?

```
int A (int n, int d){
  if(d*d>n) return 1+n;
  if(d*d==n)return d+A(n,d+1);
  if(!(n%d))return A(n,d+1);
            return d+n/d+A(n,d+1);
}
```

a) **-4**, pentru **A(-5,1);**
b) **4**, pentru **A(3,2);**
c) **6**, pentru **A(5,2);**
d) Suma divizorilor proprii, pentru **A(n,2)** cu **n>1**;

27. Ce returnează?

```
long Y(long a, long b){  //a,b>=1
  if(b>a)  return 0;
  if(b==1) return a;
  if(b<=a) return (a-b+1)*Y(a,b-1)
}
```

a) **1**, pentru **Y(1,1);**
b) **3**, pentru **Y(3,2);**
c) **20**, pentru **Y(5,2);**
d) Combinari de **a** luate cate **b**;

28. Ce returnează?

```
bool A(int  B, int C){
  while (C>0 && (B!=(C%10)))
           C/=10;
  if(C==0) return false;
           return true;
}
```

a) **false**, pentru **A(1,1);**
b) **true**, pentru **A(2,2);**
c) **false**, pentru **A(5,22);**
d) **true**, pentru **A(5,25);**

29. Ce face?

```
void X(int n,int v[]){//n lung. lui v
 for(int k=1;k<n;k++)
  for(int i=1;i<n;i++)
   { if(v[i]<v[i+1])
         Schimb(v[i],v[i+1]);//interschimba
   }
}
```

a) BubbleSort crescator vectorul **v**;
b) SelectSort descrescător vectorul **v**;
c) SelectSort crescător vectorul **v**;
d) BubbleSort descrescator vectorul **v**;

30. Ce returnează funcţia

```
long A(long n, long p){
 if(n>0) {
  if((n%10)%3==0)return A(n/10,p*10)+(9-n%10)*p;
                 return A(n/10,p);
  }
                 return 0;
}
```

a) **14**, pentru **n=147**, **p=1**;
b) **63**, pentru **n=451236**, **p=1**;
c) **36**, pentru **n=512346**, **p=1**;
d) **360**, pentru **n=632514**, **p=10**;

31. **Ackermann**.
Se consideră numerele naturale m și n (0 ≤ m ≤ 10, 0 ≤ n ≤ 10) și subalgoritmul Ack(m, n) care calculează valoarea funcției Ackermann pentru valorile m și n.

```
Subalgoritm Ack(m, n)
    Dacă m = 0
      atunci returnează n + 1
      altfel
         Dacă m > 0 și n = 0
               atunci returnează Ack(m - 1, 1)
               altfel returnează Ack(m - 1, Ack(m, n - 1))
         SfDacă
    SfDacă
SfSubalgoritm
```

Precizați de câte ori se autoapelează subalgoritmul Ack(m, n) prin executarea secvenței de instrucțiuni:

```
m ← 1, n ← 2
Ack(m, n)
```

A. de 7 ori
B. de 10 ori
C. de 5 ori
D. de același număr de ori ca și în cazul executării secvenței de instrucțiuni:

```
m ← 1, n ← 3
Ack(m, n)
```

32. Ce conţine vectorul V, după execuţia funcţiei C, n întreg.

```
void C(long n)
{ int c;
  int V[10]={0};
  while(n>0)
  { c=n%10; V[c]++;
    n/=10;
  }
}
```

a) doar cifre binare, pentru n>0;
b) doar cifre 0, pentru n=-123;
c) frecvenţa de apariţie a cifrelor, pentru n>0;
d) doar frecvenţa de apariţie a cifrei 0, pentru n>0;

33. Ce face metoda A, ştiind că X e vectorul de frecvenţă al cifrelor lui n>0.

```
bool A(long n, int X[]) {
  //se determină X
  if(X[0]>0) return false;
  for(i=1; i<=9;i++)
      if(X[i]>1) return false;
  return true;
```

a) false, dacă n conţine cifra 0;
b) true, dacă n conţine toate cifrele de la 1 la 9;
c) false dacă conţine cifra 5 de două ori;
d) true, dacă n are doar cifre nenule distincte;

34. Ce **returnează? (e de la un concurs mai vechi)**

Se consideră subalgoritmul alg(x, b) cu parametrii de intrare două numere naturale ***x*** și ***b*** ($1 \le x \le 1000$, $1 < b \le 10$).

```
Algoritmul  alg(x,b):
  s ← 0
  CâtTimp x > 0 execută
     s ← s + x MOD b
     x ← x DIV b
  SfCâtTimp
  returnează s MOD (b - 1) = 0
SfAlgoritm
```

Precizaţi efectul acestui algoritm:

a) returnează suma cifrelor reprezentării în baza b a numărului natural x;
b) returnează suma cifrelor reprezentării în baza b-1 a numărului x dacă este divizibilă cu b-1;
c) returnează true dacă numărul natural x este divizibil cu b-1;
d) returneaza true suma cifrelor reprezentării în baza b a numărului x este divizibilă cu b-1;

35. Avem 4 variante a algoritmului de căutare binară, care e corectă (x este, aprioric, sortat crescător)?

```
A.
int CBR(int Ls, int Ld, int x[], int val){
  int m;          //Ls=1, Ld=lung.lui x
  m=(Ls+Ld)/2;    //la apelul primar
  if(val==x[m]) return m;
  if(Ls<=Ld){
    if(val<x[m]) return CBR(Ls,m-1,x,val);
                 return CBR(m+1,Ld,x,val);
  }
  return -1;
```

```
B.
int CBR(int Ls, int Ld, int x[], int val){
  int m;          //Ls=1, Ld=lung.lui x
  m=(Ls+Ld)/2;    //la apelul primar
  if(val==x[m]) return m;
  if(Ls<=Ld)
    if(val<x[m]) return CBR(Ls,m+1,x,val);
    else         return CBR(m-1,Ld,x,val);
  return -1;
}
```

```
C.
int CB(int n, int x[], int val)
{ int ls,ld,m;    //n lungimea lui x
  ls=1;
  ld=n;
  while (ls<=ld)
    { m=(ls+ld)/2;
      if(val==x[m]) return m;
      if(val<x[m]) ld=m+1;
      else         ls=m-1;
    }
  return -1;
}
```

```
D.
int CB(int n, int x[], int val)
{ int ls,ld,m;  //n lungimea lui x
  ls=1;
  ld=n;
  while (ls<=ld)
    { m=(ls+ld)/2;
      if(val==x[m]) return m;
      if(val<x[m]) ld=m-1;
      else         ls=m+1;
    }
  return -1;
}
```

36. Fie Algoritmul Calcul(a,b) cu parametrii a,b numere naturale, 1 ≤ a,b ≤ 1000. (Din concurs mai vechi)

```
1.  Algoritmul Calcul (a,b)
2.    Dacă a ≠ 0
3.      atunci retur Calcul(a DIV 2, b+b) + b*(a MOD 2)
4.    SfDacă
5.    retur 0
6.  SfAlgoritm
```

Care afirmaţii sunt false?

a. Dacă a şi b sunt egale, algoritmul returnează valoarea lui a;
b. Dacă a=1000 şi b=2, algoritmul se autoapeleză de 10 ori;
c. Valoarea calculată şi returnată de algoritm este a / 2 + 2*b;
d. Instrucţiunea de pe linia 5 nu se execută niciodată;

37. Fie funcţia de mai jos. Cum se aranjează elementele vectorului X, după execuţia funcţiei.

```
void F(int n, int a[]){
  int i,j,aux;
  i=j=1;
   while(j<=n){
       if(a[j]>=0) j++;
       else { Schimb(a[j],a[i];
              i++,j++;
            }
   }
}
```

A. Primele elemente sunt >0 apoi ultimele sunt ≤0;
B. Primele elemente sunt ≤0;
C. Exista k∈{1,...,n} astfel că primele k sunt amestecate, iar ultimele (n-k) >0;
D. Exista k∈{0,...,n} astfel că primele k sunt ≤0, iar ultimele (n-k) >0;

38. Care algoritmi calculează combinări de n luate câte k.

```
A.
long A(long n, long k){//n,k naturali
   if(n<k)          return 0;
   if(k==0||k==n)   return 1;
   return n*A(n-1,k)/(n-k);
}
```

```
B.
long B(long n, long k){//n,k naturali
   if(n<k)          return 0;
   if(k==1)         return n;
     return (n-k+1)*B(n,k-1);
}
```

```
C.
long C(long n){//n natural
   if(n==0 || n==1) return 1;
            return n*C(n-1);
}
```

```
D.
long D(long n, long k){//n,k naturali
   if(n<k)          return 0;
   if(k==0||k==n)   return 1;
   if(k==1)         return n;
   return n*D(n-1,k-1)/k;
}
```

39. Ce se afişează?

```
long CeFace(long n, long z, long p){
 if(n>0) {
   if((n%10)%3==0) return CeFace(n/10, z+(9-n%10)*p, p*10);
                   return CeFace(n/10,z,p);
  }
  else             return z;
}
```

a) 631 pt. cout<<CeFace(12345,1,10);
b) 632 pt. cout<<CeFace(123456,2,10);
c) 633 pt. cout<<CeFace(1234567,3,10);
d) 634 pt. cout<<CeFace(123456789,4,10)

40. Ce face algoritmul ?

```
void PM(int x[], int n, int P[], int &k){
      //n e lungimea lui x, k lungimea lui P
   P[k=1]=1;           //k=1 si p[k]=1
   for(int i=2;i<=n;i++)
    { if(x[i]>x[P[1]]) P[k=1]=i;
      else
           if(x[i]==x[P[1]]) P[++k]=i;
    }
}
```

A. Pentru x=(9,8,7,6,5,4,3,2,1) $\Rightarrow$ P=(1,2,3);
B. Pentru x=(1,3,2,4,6,5,8,7,9) $\Rightarrow$ P=(1,3,2);
C. Pentru x=(8,4,3,5,8,2,8,5,1) $\Rightarrow$ P=(1,5,7);
D. Pentru x=(8,4,3,2,8,2,5,8,3) $\Rightarrow$ P=(4,6);

41. Ce returnează funcţia recursivă?

```
int A(long n, long d)
{ if(n<2)     return 0;
  if(d>=2){
    if(n%d==0)  return 0;
                return A(n,d-1);
  }
  return 1;
}
```

a) 0 pentru A(-5,10);
b) 0 pentru A(2,2);
c) 1 pentru A(13,13);
d) 0 pentru A(14,2);

42. **Ce valori sunt necesare?**
Se consideră subalgoritmul dif(a, n), unde a este un șir cu n numere întregi (n – număr natural, 0 < n < 100):

```
Subalgoritm dif(a, n)
     Dacă n = 0 atunci  returnează 0
     SfDacă
     Dacă |a[n]| MOD 2 = 0 // |a[n]| reprezintă valoarea absolută a lui a[n]
       atunci  returnează dif(a, n - 1) + a[n]
       altfel  returnează dif(a, n - 1) - a[n]
     SfDacă
SfSubalgoritm
```

Precizați pentru care valori ale lui n și a subalgoritmul returnează valoarea 0.

A. n = 4 și a = (6, 4, 5, 5)
B. n = 4 și a = (-6, 5, 4, -7)
C. n = 8 și a = (-6, 5, -1, -4, 1, 4, -7, 6)
D. n = 8 și a = (-6, -3, 0, 1, 2, 3, -1, 4)

43. Metoda AB apelează metoda X, verificaţi care din afirmaţiile din dreapta sunt corecte.

```
bool X(short C, long N){
  while (N>0 && (C!=(N%10)))
     N/=10;
  if(N==0) return false;
           return true;
}
bool AB(long A, long B){
    short U;
    while (A>0) {
      U = A % 10;
      if (!X(U,B)) return false;
      A/=10;
    }
    return true;
}
```

A. Apelul AB(1234, 5678) returnează **true**;
B. Apelul AB(1234, 34251) returnează **true**;
C. Apelul AB(1234, 5678) returnează **false**;
D. Apelul AB(1111,1112) returnează **true;**

44. Se dă metoda A. Ce valori au parametri de iesire B,D,C, (aprioric la apelul primar avem B=C=D=0) pentru:

```
void A(int m, int X[], int n, int Y[], int& B, int& D, int& C){
  for(int i=1;i<=m;i++){                    //B=D=C=0, la apelul primar
    for (int j=1;j<=n;j++) {
      int k=0;
      while ((i+k<=m) && (j+k<=n) && (X[i+k]==Y[j+k]))
               k++;
      if (k>D)
           { B  =i;
             D  =k;
             C  =j;
         }//end if
    }//end for
  }//end for
}//end A
```

A. m=8, X={1,4,-1,5,10,2,30,17};
n=14, Y={-2,-7,1,5,-1,7,10,2,30,25,17,5,10,2}; ⇒ B=5,D=3, C=7;
B. m=8, X={1,4,-1,5,10,2,30,17};
n=14, Y={-2,-7,1,5,-1,7,10,2,30,25,17,5,10,2}; ⇒ B=4,D=3, C=12;
C. m=5, X={1,1,1,1,1}
n=7, Y={1,1,1,1,1,1,1} ⇒ B=1,D=5,C=1;
D. m=5, X={1,1,1,1,1}
n=7, Y={1,1,1,1,1,1,1} ⇒ B=1,D=5,C=2;

45. Care afirmaţii sunt adevărate, pentru cei 4 algoritmi?

A. algoritmul **a** şi algoritmul **b** fac sortarea prin metoda bulelor;

B. Algoritmul **c** şi algoritmul **d** fac sortarea prin inserţie;

C. Algoritmul **d** face sortarea prin selecţie;

D. Algoritmul **c** face sortarea prin inserţie;

a.

```
void A(int n,int v[]){//n=lungime  v
  for(int k=1;k<n;k++){
     for(int i=1;i<n;i++){
         if(v[i]>v[i+1])
           Schimb(v[i],v[i+1]);

     }
  }
}
```

b.

```
//apelul primar cu i=1, Ok=true
void A (int n,int x[], int i, bool Ok){
  if(i<n){ if(x[i]>x[i+1]) {
              Schimb(x[i],x[i+1]);
              Ok=false;
           }
           A(n,x,i+1,Ok);
         }
   if(!Ok) A(n,x,i=1,Ok=true);
}
```

c.

```
//apelul primar cu i=2, j=1, aux=x[2]
void C(int n,int i, int j,int x[], int aux){
    if(i<=n)
       if(j>0 && aux < x[j]){
          x[j+1]=x[j];
          C(n,i,j-1,x,aux);
       }
       else { x[j+1]=aux;
              C(n,i+1,i,x,x[i+1]);
            }
}
```

d.

```
//apelul primar D(n,v,i=1,j=2)
void D(int n, int v[],int i, int j){
  if(i<n){
     if(j<=n){
               if(v[i]>v[j]) Schimb(v[i],v[j]);
               D(n,v,i,j+1);
             }
     else      D(n,v,i+1,i+2);
  }
}
```

46. Ce face programul ?

```
void P(long long n){
 long long i,j,k,p,limJ,cont;
 cont=0; i=3;
 while(cont<n){
   p=i/2;
   if(i%2==0) limJ=p*p-1;
   else       limJ=2*p*(p+1);
   j=i+1;
   while(cont<n && j<=limJ){
     k=sqrt(i*i+j*j);
     if(k*k==i*i+j*j) cout<<setw(5)<<++cont<<". "<<setw(10)<<i<<setw(10) <<j<<setw(10)<< k<<endl;
     j++;
   }//end while dupa cont si j
   i++;
 }//end while dupa cont
}//end functie
```

a) pt. n=17 se afişează:

1. 3 4 5
2. 6 8 10
3. 8 15 17
4. 9 12 15

b) pt. n=7 se afişează:

1. 3 4 5
2. 5 12 13
3. 6 8 10
4. 7 24 25
5. 8 15 17
6. 9 12 15
7. 9 40 41

c) pt. n=5 se afişează:

1. 3 4 5
2. 5 12 13
3. 7 24 25
4. 9 12 15
5. 9 40 41

d) pt. n=6 se afişează:

1. 3 4 5
2. 6 8 10
3. 8 15 17
4. 10 24 26
5. 12 16 20
6. 12 35 37

47. Ce face returnează funcţia recursivă?

```
int A(long n, long d)
{ if(n<2)     return 0;
  if(d*d<=n){
    if(n%d==0)  return 0;
      if(d==2)  return A(n,d+1);
                return A(n,d+2);
  }
  return 1;
}
```

a) 0 pentru A(-5,10);
b) 0 pentru A(2,2);
c) 1 pentru A(13,4);
d) 0 pentru A(14,2);

48. Ce face algoritmul ?

```
//x e vector, n e lungimea lui x
int Im,iM;
Im=iM=n;
for(int i=n-1;i>0;i--)
    { if(x[i]>x[Im]) Im=i;
      if(x[i]<x[iM]) iM=i;
    }
```

a) Pentru x=(9,8,7,6,5,4,3,2,1) ⇒ Im=9,iM=1;
b) Pentru x=(9,8,7,6,5,4,3,2,1) ⇒ Im=5,iM=5;
c) Pentru x=(9,8,7,6,5,4,3,2,1) ⇒ Im=1,iM=9;
d) Pentru x=(9,8,7,6,5,4,3,2,1) ⇒ Im=9,iM=9;

49. Ce se afişează?

```
long CeFace(long n, long p){
  if(n>0) {
   if((n%10)%3==0) return CeFace(n/10,p*10)+(9-n%10)*p;
      else         return CeFace(n/10,p);
  }
  else             return 0;
}
```

a) 6   pt. cout<<CeFace(12345,10);
b) 630 pt. cout<<CeFace(123456,10);
c) 6300 pt. cout<<CeFace(1234567,10);
d) 6300 pt. cout<<CeFace(123456789,10)

50.Ce face algoritmul următor?

```
void A(long n, long d){  //n,d naturali,n≥d, apel primar cu d=2
  if(d*d>n) cout<<"1,"<<n<<endl;
  else
   {  if(n%d==0)
         if(n/d==d) {cout<<d<<",";           A(n,d+1);}
         else       {cout<<d<<","<<n/d<<","; A(n,++d);}
      else A(n,d+1);
   }
}
```

A. Afişează divizorii naturali ai lui n (n>1);
B. Afiseaza divizorii proprii ai lui n (n>1);
C. Afişează un număr par de divizori ai lui n (n>1);
D. Afişează un număr impar de divizori ai lui n (n>1);

51. Ce face funcţia

```
//m,n>1 naturale si m,n <10, n2=n, la apel primar
void citR(int m, int n, int n2, int A[][10]){
   if(m>0)
     if(n>0){
        citR(m,n-1,n2,A);
        cout<<"A["<<m<<"]["<<n<<"]=";
        cin >> A[m][n];
     }
     else citR(m-1,n2,n2,A);
}
```

a) Citeşte recursiv o matrice coloană după coloană începând cu prima coloană;
b) Citeşte recursiv o matrice linie dupa linie începând cu prima linie;
c) Citeşte recursiv o matrice linie dupa linie începând cu ultima linie;
d) Citeşte recursiv o matrice coloană după coloană începând cu ultima coloană;

52. Care afirmaţie e adevărată?

```
long CeFace(long N,int L,int i,int J){//N>0,intreg
    long Change=0;
    long P10=1;
    int P=L-i+1;
    int cont=1,C;
    while (N){
        C=N%10;
        if(cont==P)
            Change=Change+J*P10;
        else
            Change=Change+C*P10;
        P10*=10;
        N/=10;
        cont++;
    }
   return Change;
}
```

a) CeFace(123,3,4,6), retur 123
b) CeFace(654321,6,3,0), retur 354321
c) CeFace(654321,6,3,1), retur 651321
d) CeFace(654321,8,3,0), retur 54321

53. Ce returnează metoda Func (vectorul F are valorile unei funcţii definite pe {1..m} cu valori in {1..n})?

```
bool esteFunctie(int m, int F[], int n){
    for(int i=1;i<=m;i++)
      if(F[i]<1 || F[i] >n) return false;
    return true;
}
bool Func (int m, int F[], int n)
{   if(m>n) return false;
    if(!esteFunctie(m,F,n)) return false;
    for(int i=1;i<=m-1;i++)
       for(int j=i+1; j<=m;j++)
          if(F[i]==F[j]) return false;
    return true;
}
```

a) true daca F nu este funcţie bijectivă.
b) true dacă F este funcţie surjectivă.
c) true dacă F este doar functie.
d) true dacă F este funcţie injectivă.

54. Ce returnează funcţia Func (vectorul F are valorile unei funcţii definite pe `{1..m}` cu valori in `{1..n}`)?

```
bool esteFunctie(int m, int F[], int n){
    for(int i=1;i<=m;i++)
      if(F[i]<1 || F[i] >n) return false;
    return true;
}
bool Exista(int m, int F[], int val){
    for(int i=1; i<=m;i++)
      if(F[i]==val) return true;
    return false;
}
bool Func (int m, int F[], int n){
    if(m<n) return false;
    if(!esteFunctie(m,F,n)) return false;
    for(int i=1;i<=n;i++)
      if(!Exista(m,F,i)) return false;
    return true;
}
```

a) true daca F nu este bijectivă
b) true dacă F este surjectivă
c) true dacă F este doar functie
d) true dacă F este injectivă

55. **Completați.**
Se consideră subalgoritmul eliminImpar(n), unde n este un număr natural, 1 ≤ n ≤ 100 000.

```
Subalgoritm eliminImpar(n)
     Dacă n = 0 atunci  returnează 0
     SfDacă
     Dacă n MOD 2 = 1 atunci  returnează  eliminImpar(n DIV 10)
     SfDacă
     returnează ...
SfSubalgoritm
```

Precizați instrucțiunea care ar trebui scrisă în locul punctelor de suspensie astfel încât algoritmul să aibă ca efect eliminarea cifrelor cu valori impare din numărul n.

A. eliminImpar(n MOD 10) * 10 + n DIV 10
B. eliminImpar(n) * 10 + n MOD 10
C. eliminImpar(n DIV 10) * 10 + n MOD 10
D. eliminImpar((n DIV 10) MOD 10) * 10

56. Se dă metoda A. Ce valori au parametri de iesire B,D,C?

```
void A(int m, int X[], int& B, int& D, int& C){
  B=C=D=0;
  for(int i=1;i<m;i++){
    for (int j=i+1;j<=m;j++) {
      int k=0;
      while ((i+k<m) && (j+k<=m) && (X[i+k]==X[j+k]))
               k++;
      if (k>D)
           { B  =i;
             D  =k;
             C  =j;
         }//end if
    }//end for
  }//end for
}//end A
```

A. m=8, X={1,4,-1,5,10,2,30,17}; ⇒ B=0,D=0, C=0;
B. m=8, X={1,4,-1,5,10,1,4,17}; ⇒ B=1,D=2, C=6;
C. m=5, X={1,1,1,1,1} ⇒ B=1,D=4,C=2;
D. m=5, X={1,1,1,1,1} ⇒ B=1,D=5,C=1;

57. Care numere naturale au un singur divizor propriu?
   A. Numerele prime;
   B. Numerele impare;
   C. Pătratele perfecte;
   D. Pătratele perfecte $n=p^2$, cu p prim;

58. Câte numere care au mai mult de 4 (>4) divizori sunt de la 1 până la n, (n≥12) ?
   A. $n-(1+|\{p/p\leq n\ si\ p\ prim\}|+\sum_{p^2\leq n,p\ prim}1+\sum_{p^3\leq n,p\ prim}1+\sum_{pq\leq n}1)$, p,q, prime, p<q.
   B. 39, dacă n=100;
   C. 38, dacă n=100;
   D. Pătratele perfecte $n=p^{2k}$, cu p prim şi k≥2;

59. Ce se află în vectorul P, de n componente, care aprioric sunt fiecare iniţializate cu 0, după execuţia codului:

```
Pt i=2,[√n] execută
  Pt k=2*i,n,i execută
        P[k]=1;
  SfPt
SfPt

A. Dacă i este par   (1 ≤ i ≤ n) atunci P[i]=1;
B. Dacă i este impar (1 ≤ i ≤ n) atunci P[i]=0;
C. Dacă i este prim  (1 ≤ i ≤ n) atunci P[i]=1;
D. Dacă i este prim  (1 ≤ i ≤ n) atunci P[i]=0;
```

60. Ce returnează metoda?

```
  long CalcRest(long i, long j, long k){
        long iModuloK=i%k;
        long R=1;
        while (j--)
              R=(R*iModuloK)%k;   ///atat lui i cat si produsului  se aplica %
        return R;
  }

A. R=2  pentru i=100,  j=100 şi k=7;
B. R=1  pentru i=1000, j=1000 şi k=7;
C. R=1  pentru i=n+1,  j oarecare şi k=n;
D. R=2  pentru i=1000, j=1000 şi k=7.
```

61. Ambele metode afişează toate tripletele (i,j,k) de numere naturale care verifică condiţiile:

$i^2+j^2 = k^2$

$i<j<k<=n$ (n dat)

1.

```
  void Pitagora(long n){
        long Cont=0;                  //Cont este contor pentru triplete
        for(long i=3;i<=n-2;i++)      //tripletele pitagorice incep cu 3,4,5
              for(long j=i+1;j<=n-1;j++)
                    for(long k=j+1;k<=n;k++)
                          if(i*i+j*j==k*k)
                                cout<<++Cont<<'.'<<i<<' '<<j<<' '<<k<<endl;

  }
```

2.

```
void Pitagora( long n){
        long   i,j;
        long   Cont=0;              //Cont este contor pentru triplete
        long   NPeRad2 =n/sqrt(2); //tripletele pitagorice incep cu 3,4,5
        for(i=3;i<=NPeRad2;i++){   //i<=[n/sqrt(2)]
                long iLa2=i*i;
                j        =i+1;
                while (j<=n-1){
                        long SumaPatrate=iLa2+j*j;
                        long k=sqrt(SumaPatrate);
                        if(k<=n && k*k == SumaPatrate)
                                cout<<++Cont<<'.'<<i<<'.'<<j<<'.'<<k<<endl;
                        j++;
                }
        }
}
```

Care afirmaţii sunt adevărate:

A. Metoda 1 este mai eficientă;
B. Metoda 2 este mai eficientă;
C. Ambele metode sunt la fel de eficiente;
D. Metoda 2 are complexitatea T(n) = O(n^2).

62. Ce returnează metoda de mai jos?

```
int  SumaDiv(long n, long d){    //d=2 la apelul primar, n>0
        if(n<=1)    return 0;
        if(d*d>n)   return 1;
        if(d*d==n)  return 1+d;
        if(n%d==0)  return d+n/d + SumaDiv(n,d+1);
                    return SumaDiv(n,d+1);
}
```

A. Suma divizorilor lui n, (n>1);
B. Suma divizorilor proprii, ai lui n, (n>1);
C. Suma divizorilor lui n fără n, (n>1);
D. Suma divizorilor improprii al lui n, (n>1);

63. Care din metodele de calcul al C_n^k (combinări de n luate câte k) sunt mai eficiente?

1.

```
long CNK1(long n, long k){
        if(k>n)            return 0;
        if(k==0 || k==n)   return 1;
        if(k==1)           return n;
                           return CNK1(n-1,k-1)+CNK1(n-1,k);
}
```

2.

```
long CNK2(long n, long k){
        if(k>n)            return 0;
        if(k==0 || k==n)   return 1;
        if(k==1)           return n;
                           return CNK2(n-1,k-1)*n/k;
}
```

3.

```
long CNK3(long n, long k){
      if(k>n)           return 0;
      if(k==n || k==0)  return 1;
      if(k==1)          return n;
                        return CNK3(n-1,k)*n/(n-k);
}
```

A. Metodele 1 şi 2.
B. Metodele 1 şi 3.
C. Metodele 2 şi 3.
D. Toate 3 sunt la fel de eficiente.

64. Se dau două metode E şi A, n>0, m>0, k valori naturale iar X,Y, Z sunt mulţimi de întregi, A apelează E.

```
int E(int n, int X[], int val){
      for(int i=1;i<=n;i++)
            if(val==X[i])  return 1;
      return 0;
}

void A(int n, int X[], int m, int Y[], int &k, int Z[]){
      k=0;
      for (int i=1;i<=n;i++)
           if(E(m,Y,X[i])) Z[++k]=X[i];
}
```

A. Dacă X={1,3,5,7,9}; Y={1,2,3,4,5} => Z={1,2,3,4,5,5,7,9};
B. Dacă X={3,7,9,5,1}; Y={1,2,3,4,5} => Z={3,5,1};
C. Dacă X={1,3,5,7,9}; Y={1,2,3,4,5} => Z={7,9};
D. Dacă X={1,3,5}; Y={4,5} => Z={(1,4),(1,5),(3,4),(3,5),(5,4),(5,5)};

65. Se dau două metode E şi B, n>0, m>0, k valori naturale iar X,Y, Z sunt mulţimi de întregi.

```
int E(int n, int X[], int val){
      for(int i=1;i<=n;i++)
            if(val==X[i])  return 1;
      return 0;
}

void B(int n, int X[], int m, int Y[], int &k, int Z[]){
      k=0;
      for(int i=1;i<=n;i++)
            Z[++k]=X[i];
      for (int i=1;i<=m;i++)
            if(!E(n,X,Y[i])) Z[++k]=Y[i];
}
```

A. Dacă X={3,5,1,7,9}; Y={1,4,3,2,5} => Z={3,5,1,7,9,4,2};
B. Dacă X={1,3,5,7,9}; Y={1,2,3,4,5} => Z={1,3,5};
C. Dacă X={1,3,5,7,9}; Y={1,2,3,4,5} => Z={7,9};
D. Dacă X={1,3,5}; Y={4,5} => Z={(1,4),(1,5),(3,4),(3,5),(5,4),(5,5)};

66. Se dau două metode E şi C, n>0, m>0, k valori naturale iar X,Y, Z sunt mulţimi de întregi.

```
int E(int n, int X[], int val){
      for(int i=1;i<=n;i++)
            if(val==X[i])  return 1;
      return 0;
}
void C(int n, int X[], int m, int Y[], int &k, int Z[]){
      k=0;
      for (int i=1;i<=n;i++)
            if(!E(m,Y,X[i])) Z[++k]=X[i];
}
```

A. Dacă X={9,3,7,5,1}; Y={1,2,3,4,5} => Z={1,2,3,4,5,7,9};
B. Dacă X={3,5,7,9,1}; Y={1,2,3,4,5} => Z={3,5,1};
C. Dacă X={7,1,5,9,3}; Y={4,2,1,5,3} => Z={7,9};
D. Dacă X={1,3,5}; Y={4,5} => Z={(1,4),(1,5),(3,4),(3,5),(5,4),(5,5)};

67. Ce Face metoda următoare? (n,k naturale, X, P tablouri de întregi)

```
void PMN(int n, int X[],int &k, int P[]){
  int i=1;
  while(i<=n && X[i]>=0)
        i++;
  if(i≤n){    P[k=1]=i;
              for(int j=i+1;j<=n;j++){
                    if(X[j]==X[P[1]]) P[++k]=j;
                    else  if(X[j]<0 && X[j]>X[P[1]]) P[k=1]=j;
         }
  else k=0;
}
```

A. Determină poziţiile numerelor negative din X sau k=0(în caz contrar);
B. Determină valorile numerelor negative din X sau k=0 (în caz contrar);
C. Determină poziţiile numărului maxim negativ din X, sau k=0(în caz contrar);
D. Determină valoarea numărului maxim negativ din X, sau k=0 (în caz contrar);

68. Ce conţine vectorul Y (n natural, lungimea vectorului X). CreareY apelează MM.

```
int MM(int n, int X[], int i)
      {  int j=1,cnt=0;
         while(j<i)
            {if(X[j] <X[i]) cnt++;
             j++;
            }
         return cnt;
      }

void CreareY(int n, int X[], int Y[])
{ for(int i=1;i<=n;i++)
      Y[i]=MM(n,X,i);
}
```

A. X={1,2,3,4,5,6}   => y={0,1,2,3,4,5};
B. X={1,2,3,4,5,6}   => y={5,4,3,2,1,0};
C. X={1,20,3,40,5,6} => y={0,1,1,3,2,3};
D. X={1,2,3,4,5,6}   => y={1,1,1,1,1,1};

69. Metoda CreY apelează, celelalte 2 metode; m,n>0 naturale; vectorul X şi lungimea lui n, sunt date; trebuie creat vectorul Y şi lungimea lui m.

```
int ExistaInY(int m, int Y[], int v)
{   for (int i=1;i<=m; i++)
      if(v==Y[i]) return i;
    return 0;
}
void AdaugInY(int& m, int Y[], int v)
{ int j=m;
  while(j>0 && v<Y[j])
  {   Y[j+1]=Y[j];
      j--;
  }
  Y[j+1]=v;
  m++;
}
void CreY(int n, int X[], int& m, int Y[])
 { m=0;
   for(int i=1;i<=n;i++)
    { int poz=ExistaInY(m,Y,X[i]);
      if(poz==0) AdaugInY(m,Y,x[i]);
    }
 }
```

A. Dacă X={1,2,3,4,5,1,2,3,4}, n=9 => Y={5,4,3,2,1} şi m=5;
B. Dacă X={1,2,30,2,5,1,2,5,4},n=9 => Y={1,2,30,5,4} şi m=5;
C. Dacă X={1,2,30,2,5,1,2,5,4},n=9 => Y={1,2,4,5,30} şi m=5;
D. Dacă X={1,1,1,1,1,1}, n=6 => Y={1} şi m=1;

70. Metoda CMLS apelează metoda S; se dau:vectorul X şi n>0 lungimea lui X; St şi Dr trebuie determinate.

```
void S (int i, int& j, int n, int X[]){
      j=i;
      while(j<n && X[j]<X[j+1]) j++;
}
void CMLS(int& St, int& Dr, int n, int X[]){
      int i=1,j; St=0;Dr=0;
      while(i<n){
            while(i<n && X[i]>=X[i+1]) i++;
            if(i<n){   S(i,j,n,X);
                       if(j-i>Dr-St){ St=i;
                                      Dr=j;
                       }
             i=j+1;
            }
      }
}
```

A. Dacă X={1,2,3,-5,5,1,2,3,4},n=9 ⇒ St=1, Dr=5
B. Dacă X={1,2,30,2,5,1,2,5,10},n=9 ⇒ St=6, Dr=9;
C. Dacă X={1,2,30,2,5,1,2,5,4},n=9 ⇒ St=0, Dr=0;
D. Dacă X={1,1,1,1,1,1}, n=6 ⇒ St=0, Dr=0;

71. Metoda DP apelează metoda S; se dau:vectorul X şi n>0 lungimea lui X;
Vectorul P trebuie determinat.

```
void S(int &a, int &b){
      a=a*b;
      b=a/b;
      a=a/b;
}
void DP (int n, int X[], int P[]){
      for(int i=1;i<=n;i++) P[i]=i;
      for(int i=1;i<n;i++)
            for (int j=i+1; j<=n;j++)
                  if(X[P[i]] < X[P[j]]) S(P[i],P[j]);
}
```

A. Dacă X={1,2,3,-5,6,-7},n=6 => P={1,2,3,4,5,6}
B. Dacă X={1,2,30,2,8,1}, n=6 => P={3,5,4,2,1,6};
C. Dacă X={1,2,30,2,5,1}, n=6 => P={1,6,2,4,5,3};
D. Dacă X={1,1,1,1,1,1}, n=6 => P={1,2,3,4,5,6};

72. Metoda FDA schimbă ordinea elementelor iniţiale din vectorul X de lungime n, apelând metoda S; A este dat de asemenea.

```
void S(int &a, int &b){
      int   aux=a;
            a=b;
            b=aux;
}
void FDA(int n, int A,int X[])  {
    int i=1,j=1;
    while(j<=n)
       { if(X[j]<A)
            { S(X[j],X[i]);
              i++;
            }
         j++;
       }
}
```

A. Dacă X={1,2,3,-5,6,-7},n=6,A=4 => X={1,2,3,-5,-7,6}
B. Dacă X={1,2,30,2,5,1}, n=6,A=4 => X={1,2,2,1,5,30};
C. Dacă X={1,2,30,2,5,1}, n=6,A=-4 => X={2,1,30,5,2,1};
D. Dacă X={1,2,5,9,10,20},n=6,A=5 => X={20,1,2,5,9,10};

73. Metoda EA modifică vectorul X de lungime n; A este dat de asemenea.

```
int EA(int &n, int A, int X[]){
      int  i=1;
      while(i<=n)
            if(X[i]<=A){
                  for(int j=i;j<n;j++)
                        X[j]=X[j+1];
                  n--;
            }
            else i++;

}
```

A. Dacă X={1,2,3,-5,6,-7},n=6,A=4 => X={1,2,3,-5,-7}, n=5
B. Dacă X={1,2,30,2,5,1}, n=6,A=2 => X={30,5}, n=2;
C. Dacă X={1,2,30,2,5,1}, n=6,A=-4 => X={2,1,30,5,2,1},n=6;
D. Dacă X={1,2,5,9,10,20},n=6,A=5 => X={20,1,2,5,9,10},n=6;

74. Metoda trebuie să afişează soluţiile problemei $\overline{abc} = a^3+b^3+c^3$, a,b,c cifre. Ce instrucţiune trebuie pusă în locul celor 3 puncte, pentru cifra b.

```
void SumaCub(){
    int a,b,c;
    for(int i=100;i<=999;i++){
      c = i%10;
      ...
      a =(i/100);
      if(a*a*a+b*b*b+c*c*c==i) cout<<i<<endl;
    }
}
```

A. b=(i/10)/10;
B. b=(i%10)/10;
C. b=(i/100)%10;
D. b=(i/10)%10;

75. Se dă o variantă a tehnicii bulelor de sortare a vectorului V de lungime n.

```
void S(int &a, int &b){
      a=a*b; b=a/b; a=a/b;
}
void B3Cresc (int n,int v[]){
   bool ok;
   int poz=n;               //poz = pozitia ultimei schimbari dupa for
   do
   { ok=true;
     int pI=1;              //pI = pozitia unei schimbari
     for(int i=1;i<poz;i++) {
        if(v[i]>v[i+1]) {  S(v[i],v[i+1]);
                           ok=false;
                           pI=i;
                        }
     }
     poz=pI;
   }
   while (!ok);
}
```

De câte ori se apelează metoda S, pentru n=8 şi v={3,-9,123,17,-333,-56,7,27};

A) de 12 ori
B) de 13 ori
C) de 14 ori
D) de 15 ori

76. Se dă o variantă a tehnicii selecţiei de sortare a vectorului V de lungime n.

```
void S(int &a, int &b){
      a=a*b; b=a/b; a=a/b;
}
void S2Cresc(int n,int v[]){
    for(int i=1;i<n;i++){
        int pMin=i;
        for(int j=i+1;j<=n;j++){
            if(v[j]<v[pMin]) {pMin=j;}
        }
        S(v[i],v[pMin]);
    }
}
```

De câte ori se apelează metoda S, pentru n=8 şi v={3,-9,123,17,-333,-56,7,27};

A) de 6 ori
B) de 7 ori
C) de 8 ori
D) de 9 ori

77. Se dă o variantă a tehnicii inserţiei de sortare a vectorului V de lungime n.

```
void ICresc(int n,int v[]){
    for(int i=2;i<=n;i++){
        int aux=v[i];
        int j=i-1;
        while(j>0 && aux< v[j]){
            v[j+1]=v[j];            (*)
            j--;
        }
        v[j+1]=aux;
    }
}
```

De câte ori se execută instrucţiunea (*) pentru n=8 şi v={3,-9,123,17,-333,-56,7,27};

A) de 12 ori
B) de 13 ori
C) de 14 ori
D) de 15 ori

78. Se dau 2 numere naturale a, b, astfel ca 1<a<b. Metoda C apelează d2p, iar d2p metoda P. Metoda P determina primalitatea unui numar natural. Parametri a şi b sunt daţi de asemenea.

```
long P(long k)
{   ...
}
int d2p(long x, long &p, long &q)
{ long d=2;
  while(d*d<=x)
  { if(x==d*(x/d))
      { if(P(d) && P(x/d))
          {   p=d;
              q=x/d;
              return 1;
          }
      }
      if(d==2)d=3;
      else    d+=2;
  }
  return 0;
}
void C(long a, long b)
{   long x=a,p1,p2,p3,p4;
    while(x<b)
    {if(d2p(x,p1,p2) && d2p(x+1,p3,p4))
        {   cout<<x<<"="<<p1<<"*"<<p2<<" ";
            cout<<x+1<<"="<<p3<<"*"<<p4<<endl;
        }
        x++;
    }
}
```

Ce se afişează?

A. Pentru a=115 şi b=125
```
            118=2*59  119=7*17
            121=11*11 122=2*61
            122=2*61  124=2*62
```
B. Pentru a=115 şi b=125
```
            118=2*59  119=7*17
            121=11*11 122=2*61
            122=2*61  123=3*41
```
C. Pentru a=30 şi b=40
```
            33=3*11  34=2*17
            34=2*17  35=5*7
            37=1*37  38=2*19
```

D. Pentru a=30 şi b=40
```
            33=3*11  34=2*17
            34=2*17  35=5*7
            38=2*19  39=3*13
```

79. Metoda următoare, afişează triplete „heronice”. Un triplet (i,j,k) de numere naturale se numeşte „heronic”, dacă îndeplineşte condţiile:

1) i,j,k sunt măsurile laturilor unui triunghi, i≤j<k.

2) aria triunghiului se exprimă printr-un număr natural de unităţi.

```
void Heron(long n){
   long k,rad,cont=0;
   for(long i=3;i<n;i++)
    for(long j=i;j<n;j++)
      for(long k=j+1;k<=n;k++){
          if(i>k-j)           //verificarea ca e trinughi
                              //verificare arie numar natural;
                { double p=(i+j+k)/2.0;
                  double ALaPatrat=p*(p-i)*(p-j)*(p-k);
                  rad    =sqrt(ALaPatrat);
                  if(rad*rad==ALaPatrat)
                   {cout<<setw(5)<<++cont<<". "<<setw(5)<<i<<setw(5)<<j<<setw(5)<<k;
                    cout<<" Aria="<<setw(5)<<rad<<endl;
                   }
                }
      }
}
```

A. dacă n=10 atunci se afişează:

```
1.   3   4    5 Aria=6
2.   5   5    6 Aria=12
3.   5   5    7 Aria=13
4.   5   5    8 Aria=12
```

B. dacă n=10 atunci se afişează:

```
1.   3   4    5 Aria=6
2.   5   5    6 Aria=12
3.   5   5    8 Aria=12
4.   5   5   10 Aria=20
```

C. Dacă n=10 atunci se afişează:

```
1.   3   4    5 Aria=6
2.   5   5    6 Aria=12
3.   5   5    8 Aria=12
4.   6   8   10 Aria=24
```

D. Toate tripletele pitagorice sunt şi „heronice”.

80. Metoda CreV, crează vectorul V cu numere naturale, n este data de intrare.

```
void CreV(int n, int V[]){
    int N;
    int k;
    int i;
    V[1]=N=k=1;
    while(k<n){
        N++;
        i=1;
        while(k<n && i<=N)
            V[++k]=i++;
    }
}
```

Ce valori are vectorul V?

A. dacă n=10 atunci V={1,1,1,1,1,1,1,1,1,1}
B. dacă n=10 atunci V={1,2,3,4,5,6,7,8,9,10}
C. dacă n=10 atunci V={1,1,1,2,1,3,1,4,1,5}
D. dacă n=10 atunci V={1,1,2,1,2,3,1,2,3,4}

81. Metoda Rez, crează vectorul V cu numere naturale, n este dată de intrare. Metoda Rez apelează P.

```
int  P(long n){
  if(n<2) return 0;
  if(n>2 && n%2==0) return 0;
  for(int d=3;d*d<=n;d+=2)
       if(n%d==0) return 0;
  return 1;
}
void Rez(int n,int V[])
{   int p=3,nr=0;
    while(nr<n)
    { if(P(p) && P(p+2))
        {   V[++nr]=p;
            V[++nr]=p+2;
        }
        p=p+2;

    }
}
```

Ce conţine vectorul V?

A. n=8 ⇒ V={1,2,3,5,7,11,13,17}
B. n=8 ⇒ V={3,5,7,11,13,17,19,23}
C. n=8 ⇒ V={3,5,5,7,11,13,17,19}
D. n=8 ⇒ V={3,5,7,11,13,17,19,21}

82. Metoda Rezolv, crează vectorul x cu numere naturale, n este data de intrare. Rezolv apelează Prim care determină primalitatea unui numar natural.

```
void Rezolv(int n, int x[])
{   int i,j,k;
    x[1]=1;
    i=k=2;
    while(i<=n)
    { if(Prim(k))
        {   j=1;
            while(i<=n && j<=k)
                x[i++]=j++;
        }
        else{   j=1;
                while(i<=n && j<=k)
                    {  x[i++]=k;
                       j++;
                    }
            }
        k++;
    }
}
```

Ce conţine vectorul x?

A. n=12 ⇒ x={1,2,3,1,2,3,4,4,4,4,1,2};

B. n=12 ⇒ x={1,1,2,1,2,1,2,3,4,4,4,4};

C. n=12 ⇒ x={1,1,2,1,2,3,4,4,4,4,1,2};

D. n=12 ⇒ x={1,1,2,1,2,3,3,3,4,4,4,4}.

83. Metoda Comp compune doi vectori ce reprezinta 2 funcţii, F şi G. Ce conţine vectorul GF; n lungimea lui F, precum F şi G sunt date.

```
void Comp(int n, int F[], int G[], int GF[])
{   for(int i=1;i<=n;i++)
        GF[i]=G[F[i]];
}
```

A. n=4, F=[2,1,3,3], G=[2,1,4] ⇒ GF=[1,2,3,4]

B. n=5, F=[5,1,2,3,4] G=[1,1,1,2,3] ⇒ GF=[4,1,1,2,3]

C. n=4, F=[2,1,3,3], G=[2,1,4] ⇒ GF=[1,2,4,4]

D. n=5, F=[5,1,2,3,4] G=[1,1,1,2,3] ⇒ GF=[3,1,1,1,2]

84. Se dă o matrice A de cel mult 20 de linii si coloane. Se mai dau m-numarul de linii efectiv şi n-numărul de coloane efectiv; m,n din {1,2,3,4,5,...19};
Ce valori au parametri de iesire L şi C, după execuţia metodei următoare:

```
void PM(int m, int n, int A[20][20], int& L, int& C){
    L=-1, C=-1;
    for(int i=0;i<m;i++)
        for(int j=0;j<n;j++)
            {if(A[i][j]>=0 && C==-1)
                 {L=j;
                  C=i;
                 }
             else
                 if(A[i][j]>=0 && A[i][j]<A[C][L] )
                     {L=j;
                      C=i;
                     }
            }
}
```

A. **L şi C** conţin numărul coloanei, respectiv al liniei în care se găseşte valoarea maximă din matrice;
B. **L şi C** conţin numărul liniei, respectiv al coloanei în care se găseşte valoarea maximă din matrice;
C. **L şi C** conţin numărul coloanei, respectiv al liniei în care se găseşte valoarea minimă pozitivă din matrice;
D. **L şi C** conţin numărul liniei, respectiv al coloanei în care se găseşte valoarea minimă pozitivă din matrice;

85. Ce afişează următoarea secvenţă de instrucţiuni în pseudocod?

```
citeşte n (număr natural)
k←1
┌cât timp n≥1 execută
│ ┌dacă n>k
│ │  atunci i←k
│ │  altfel i←n
│ └■
│  n←n-i
│ ┌cât timp i≥1 execută
│ │  scrie k,',';
│ │  i←i-1
│ └■
│  k←k+1
└■
```

A. n=9, ⇒ 1,2,2,3,3,4,4,5,5;
B. n=10, ⇒ 1,1,2,2,3,3,4,4,5,5;
C. n=11 ⇒ 1,2,1,2,3,1,2,3,4,1,2;
D. n=12 ⇒ 1,2,2,3,3,3,4,4,4,4,5,5.

86. Se dă o matrice A pătratică de ordinul n, n>1. Matricea este booleană (conţine doar valori de 0,1). Ştiind că numărul de 1 este două treimi din numărul de 0, aflaţi ce valori poate lua n.

A.n=4;
B.n=10;
C.n=12;
D.n=15.

87. O matrice A pătratică de ordinul n,(n>1), este booleană (conţine doar valori de 0,1) şi simetrică faţă de prima diagonală. Indicaţi numărul de matrice care se pot construi:

A.n=2 ⇒ 512 matrice;
B.n=3 ⇒ 81 matrice;
C.n=4 ⇒ 1024 matrice;
D.n=5 ⇒ 625 matrice.

88. Cei n participanţi la o competiţie sunt organizaţi în m echipe, fiecare participant facând parte dintr-o singură echipă. În timpul competiţiei fiecare participant dintr-o echipă devine prieten cu fiecare din ceilalţi participanţi din aceeaşi echipă. Dându-se m şi n aflaţi numărul maxim şi numărul minim de prietenii potenţiale.
Exemplu n = 6 participanţi şi m = 3 echipe ⇒ Max=6 pentru distribuţia 1,1,4 iar Min=3 pentru distribuţia 2,2,2.

A. n =11, m = 3 ⇒ Max=36 iar Min=15
B. n =12, m = 4 ⇒ Max=36 iar Min=10
C. n =50, m =20 ⇒ Max=190 iar Min=30
D. n =15, m = 6 ⇒ Max=45 iar Min=12

89. Ce elemente are matricea pătratică de ordinul n(linie cu linie) dupa execuţia următoarei funcţii:

```
void S(int n, int a[10][10])
{   int k=0;
    for(int i=1;i<=n/2+1;i++)
    { for(int j=i;j<=n-i+1;j++)
            a[i][j]=++k;
      for(int j=i+1;j<=n-i+1;j++)
            a[j][n-i+1]=++k;
      for(int j=n-i;j>=i;j--)
            a[n-i+1][j]=++k;
      for(int j=n-i;j>=i+1;j--)
            a[j][i]=++k;
    }
}
```

A. n =5, 1,2,3,4,5, 6,7,8,9,10, 11,12,13,14,15, 16,17,18,19,20, 21,22,23,24,25.
B. n =5, 1,2,3,4,5, 10,9,8,7,6, 11,12,13,14,15, 20,19,18,17,16, 21,22,23,24,25.
C. n =5, 1,6,11,16,21, 2,7,12,17,22, 3,8,13,18,23, 4,9,14,19,24, 5,10,15,20,25.
D. n =5, 1,2,3,4,5, 16,17,18,19,6, 15,24,25,20,7, 14,23,22,21,8, 13,12,11,10,9.

90. Metoda CreV, crează vectorul V cu numere naturale, n este data de intrare.

```
void CreV (int n, int V[]){
    int i,j,k;
    V[1]=1;
    i=k=2;
    while(i<=n)
    {   j=2;
        V[i++]=k;
        while(i<=n && j<=k/2){
                if(k%j==0)
                {   V[i]=j;
                    i++;
                }
                j++;
            }
        k++;
    }
}
```

Care din afirmaţii este corectă:

A. n=12 => V={1,2,3,4,5,6,7,8,9,10,11,12}
B. n=14 => V={1,1,2,1,3,1,2,4,1,5,1,2,3,6}
C. n=13 => V={1,2,3,2,4,5,2,3,6,7,2,4,8}
D. n=15 => V={1,2,3,4,2,5,6,2,3,7,8,2,4,9,3}

91. Metoda CreV, crează vectorul V cu numere naturale, n este data de intrare.

```
void CreV (int n, int V[]){
    int i,j,k;
    V[1]=1;
    i=k=2;                      ///i indice penru V
    while(i<=n)
    {   j=2;
        V[i++]=k;
        int U=0;
        while(i<=n && j<=k/2){
                if(k%j==0)
                {   V[i++]=j;
                    if(U==0) U=k/j;
                }
                j++;
            }
        j=2;
        while(i<=n && j<=U){
            V[i++]=U;
            j++;
        }
        k++;
    }
}
```

Care din afirmaţii este corectă:

A. n=12 => V={1,2,3,4,5,6,7,8,9,10,11,12}
B. n=14 => V={1,1,2,1,3,1,2,4,1,5,1,2,3,6}
C. n=16 => V={1,2,3,4,2,5,6,2,3,3,7,8,2,4,4,4}
D. n=19 => V={1,2,3,4,2,2,5,6,2,3,3,3,7,8,2,4,4,4,4}

92. Metoda CreV, crează vectorul V cu numere naturale, n este data de intrare.

```
int  P(long n){
        if(n<2) return 0;
        if(n>2 && n%2==0) return 0;
        for(int d=3;d*d<=n;d+=2)
            if(n%d==0) return 0;
        return 1;
      }
void CreV (int n, int V[]){
    int i,j,k;
    V[1]=1;
    i=2;
    k=2;
    while(i<=n)
    {   j=2;
        if (P(k)) V[i++]=k;
        while(i<=n && j<=k/2){
                if(k%j==0)
                {   V[i++]=j;
                }
                j++;
            }
        k++;
    }
}
```

Care din afirmaţii este corectă:

A. n=12 => V={1,2,3,2,5,2,3,7,2,4,3,5}
B. n=14 => V={1,2,3,2,5,2,3,7,1,4,3,2,3,6}
C. n=16 => V={1,2,3,2,5,2,3,7,2,4,3,2,5,11,2,3}
D. n=20 => V={1,2,3,2,5,2,3,7,2,4,3,2,5,11,2,3,4,6,13,2}

93. Metoda CreMat, iniţilizează matricea A[n][n], 2≤n≤10. Ce elemente are matricea A (linie cu linie), n este data de intrare.

```
void CreMat(int n, int A[10][10])
{ int i,j,k;
  k=0;
  for(j=1;j<=n;j++)
  {   for(i=1;i<=j;i++)
          A[i][j]=++k;
      for(i=j-1;i>=1;i--)
          A[j][i]=++k;
  }
}
```

A. n =5, 1,2,3,4,5, 6,7,8,9,10, 11,12,13,14,15, 16,17,18,19,20, 21,22,23,24,25.
B. n =5, 1,2,3,4,5, 10,9,8,7,6, 11,12,13,14,15, 20,19,18,17,16, 21,22,23,24,25.
C. n =5, 1,2,5,10,17 4,3,6,11,18, 9,8,7,12,19, 16,15,14,13,20, 25,24,23,22,21.
D. n =5, 1,2,3,4,5, 16,17,18,19,6, 15,24,25,20,7, 14,23,22,21,8, 13,12,11,10,9.

94. **Metoda CreMat, iniţilizează matricea A[n][n], 2≤n≤10. Ce elemente are matricea A (linie cu linie), n este data de intrare.**

```
void CreMat(int n, int A[10][10])
{ int i,j,k;
  k=0;
  for(i=1;i<=n;i++)
  {   for(j=1;j<=i;j++)
          A[i][j]=++k;
      for(j=i-1;j>=1;j--)
          A[j][i]=++k;
  }
}
```

A. n =5, 1,2,3,4,5, 6,7,8,9,10, 11,12,13,14,15, 16,17,18,19,20, 21,22,23,24,25.
B. n =5, 1,2,3,4,5, 10,9,8,7,6, 11,12,13,14,15, 20,19,18,17,16, 21,22,23,24,25.
C. n =5, 1,2,5,10,17 4,3,6,11,18, 9,8,7,12,19, 16,15,14,13,20, 25,24,23,22,21.
D. n =5, 1,4,9,16,25, 2,3,8,15,24, 5,6,7,14,23, 10,11,12,13,22, 17,18,19,20,21.

95. **Metoda CreMat, iniţilizează matricea A[n][n], 2≤n≤10. Ce elemente are matricea A (linie cu linie), n este data de intrare.**

```
void CreMat(int n, int A[10][10])
{ for(int i=1;i<=n;i++)
    for(int j=1;j<=n;j++)
      if(i<=j) A[i][j]=(j-1)*(j-1)+i;
      else     A[i][j]=i*i-j+1;

}
```

A. n =5, 1,2,3,4,5, 6,7,8,9,10, 11,12,13,14,15, 16,17,18,19,20, 21,22,23,24,25.
B. n =5, 1,2,3,4,5, 10,9,8,7,6, 11,12,13,14,15, 20,19,18,17,16, 21,22,23,24,25.
C. n =5, 1,2,3,4,5, 16,17,18,19,6, 15,24,25,20,7, 14,23,22,21,8, 13,12,11,10,9.
D. n =5, 1,2,5,10,17 4,3,6,11,18, 9,8,7,12,19, 16,15,14,13,20, 25,24,23,22,21.

96. **Metoda CreMat, iniţilizează matricea A[n][n], 2≤n≤10. Ce elemente are matricea A (linie cu linie), n este data de intrare.**

```
void CreMat(int n, int A[10][10])
{ for(int i=1;i<=n;i++)
    for(int j=1;j<=n;j++)
      if(j<=i) A[i][j]=(i-1)*(i-1)+j;
      else     A[i][j]=j*j-i+1;

}
```

A. n =5, 1,2,3,4,5, 6,7,8,9,10, 11,12,13,14,15, 16,17,18,19,20, 21,22,23,24,25.
B. n =5, 1,2,3,4,5, 10,9,8,7,6, 11,12,13,14,15, 20,19,18,17,16, 21,22,23,24,25.
C. n =5, 1,2,5,10,17 4,3,6,11,18, 9,8,7,12,19, 16,15,14,13,20, 25,24,23,22,21.
D. n =5, 1,4,9,16,25, 2,3,8,15,24, 5,6,7,14,23, 10,11,12,13,22, 17,18,19,20,21.

97. Metoda CreMat, iniţilizează matricea A[n][n], 2≤n≤10. Ce elemente are matricea A (linie cu linie), n este data de intrare.

```
void CreMat(int n, int A[10][10])
{ int k=0;
  for(int d=1;d<=n;d++)
    for(int j=1;j<=d;j++)
         A[d+1-j][j]=++k;
  for(int d=2;d<=n;d++)
     for(int j=d;j<=n;j++)
         A[n+d-j][j]=++k;

}
```

A. n =5, 1,2,3,4,5, 6,7,8,9,10, 11,12,13,14,15, 16,17,18,19,20, 21,22,23,24,25.
B. n =5, 1,2,3,4,5, 10,9,8,7,6, 11,12,13,14,15, 20,19,18,17,16, 21,22,23,24,25.
C. n =5, 1,3,6,10,15, 2,5,9,14,19, 4,8,13,18,22, 7,12,17,21,24, 11,16,20,23,25.
D. n =5, 1,4,9,16,25, 2,3,8,15,24, 5,6,7,14,23, 10,11,12,13,22, 17,18,19,20,21.

98. Metoda CreMat, iniţilizează matricea A[n][n], n≤10. Ce elemente are matricea A (linie cu linie), n este data de intrare.

```
void CreMat(int n, int A[10][10])
{ int k=0;
  for(int d=1;d<=n;d++)
    for(int i=1;i<=d;i++)
         A[i][d+1-i]=++k;

  for(int d=2;d<=n;d++)
     for(int i=d;i<=n;i++)
        A[i][n+d-i]=++k;

}
```

A. n =5, 1,2,3,4,5, 6,7,8,9,10, 11,12,13,14,15, 16,17,18,19,20, 21,22,23,24,25.
B. n =5, 1,2,3,4,5, 10,9,8,7,6, 11,12,13,14,15, 20,19,18,17,16, 21,22,23,24,25.
C. n =5, 1,2,5,10,17 4,3,6,11,18, 9,8,7,12,19, 16,15,14,13,20, 25,24,23,22,21.
D. n =5, 1,2,4,7,11, 3,5,8,12,16, 6,9,13,17,20, 10,14,18,21,23, 15,19,22,24,25.

99. Se dă următoarea porţiune de cod în C++ (legat de limbaj). Care afirmaţii sunt false?

```
void Exercitiu(int n){
    for(int i=1;i<=n;i++)
       int j=i*i;

}
```

A. La sfârşitul lui for,j are valoarea $(n-1)^2$;
B. La sfârşitul lui for,j are valoarea n^2;
C. La sfârşitul lui for,j are valoarea $(n+1)^2$.
D. Toate din afirmaţiile A., B., C. sunt false.

100.Fie metoda M(funcţia C+), care răspunsuri sunt corecte?

```
void M(int m, int x[], int n, inty[], int& k, int z[]){
    int i,j;
    i=j=1;
    k=0;
    while(i<=n && j<=m){
      if(x[i]<y[j])            z[++k]=x[i++];
      else if(x[i]>y[j])       z[++k]=y[j++];
           else {
                 i++;
                 j++;
                }
    }
    while (i<=m)        z[++k]=x[i++];
    while (j<=n)        z[++k]=y[j++];
 }
```

a) m=11, x={-1,2,3,41,51,61,71,81,91,301,401}; n=9, y={1,21,41,86,87,88,91,92,401};
atunci z={-1,1,2,3,41,41,51,61,71,81,86,87,88,91,91,92,301,401,401}

b) m=11, x={-1,2,3,41,51,61,71,81,91,301,401}; n=9, y={1,21,41,86,87,88,91,92,401};
atunci z={-1,1,2,3,41,41,51,61,71,81,86,87,88,91,91,92,301,401}

c) m=11, x={-1,2,3,41,51,61,71,81,91,301,401}; n=9, y={1,21,41,86,87,88,91,92,401};
atunci z={-1,2,3,21,51,61,71,81,86,87,88,92,301}

d) m=11, x={-1,2,3,41,51,61,71,81,91,301,401}; n=9, y={1,21,41,86,87,88,91,92,401};
atunci z={-1,1,2,3,41,51,61,71,81,86,87,88,91,92,301,401}

101. Fie algoritmul:

```
citeşte m,n,x (numere natural nenule, m<n)
p←0
┌cât timp m<n şi p=0 execută
│┌dacă m%x=0 şi n%x=0 atunci
│|   p←x
│|altfel
│|  ┌dacă m%x=0 atunci
│|  |       n←n-1
│|  |altfel
│|  |       m←m+1
│|  └■
│└■
└■
scrie m,' ',n
```

Care afirmaţie e adevărată?

A. m=30, n=40, x=17, se afişează 18 19

B. m=10, n=40, x=17, se afişează 18 20

C. m=11, n=30, x=7, se afişează 14 28

D. m=14, n=15, x=5, se afişează 15 15

102. Avem 9 monede de aceeaşi formă şi culoare. 8 monede au aceeaşi masă, iar una are cu 1 gram mai puţin (nu ştim care monedă). Dispunem de o balanţă(cântar) cu 2 talere. Pe fiecare taler al balanţei se pot pune oricâte monede. Care este numărul minim de utilizări ale balanţei (în orice caz) pentru a afla care este moneda cu masa mai mică.

A) 4

B) 1

C) 2

D) 3

103. Fie metoda M(funcţia C+), care răspunsuri sunt corecte?

```
void M(int m, int x[], int n, int y[], int& k, int z[]){
   int i,j;
    i=j=1;
    k=0;
    while(i<=m && j<=n){
      if(x[i]<y[j])            z[++k]=x[i++];
      else if(x[i]>y[j])       z[++k]=y[j++];
           else {
                 z[++k]=x[i++];
                 j++;
                }
    }
    while (i<=m)        z[++k]=x[i++];
    while (j<=n)        z[++k]=y[j++];
}
```

a) m=11, x={-1,2,3,41,51,61,71,81,91,301,401}; n=9, y={1,21,41,86,87,88,91,92,401};
atunci z={-1,1,2,3,21,41,41,51,61,71,81,86,87,88,91,91,92,301,401,401}

b) m=11, x={-1,2,3,41,51,61,71,81,91,301,401}; n=9, y={1,21,41,86,87,88,91,92,401};
atunci z={-1,1,2,3,21,41,41,51,61,71,81,86,87,88,91,91,92,301,401}

c) m=11, x={-1,2,3,41,51,61,71,81,91,301,401}; n=9, y={1,21,41,86,87,88,91,92,401};
atunci z={-1,1,2,3,21,41,51,61,71,81,86,87,88,91,92,301,401,401}

d) m=11, x={-1,2,3,41,51,61,71,81,91,301,401}; n=9, y={1,21,41,86,87,88,91,92,401};
atunci z={-1,1,2,3,21,41,51,61,71,81,86,87,88,91,92,301,401}

104.Ce conţine vectorul v după execuţia codului C+:

```
 void creV(int n, int V[]){
   int N, k,i;
   V[1]=N=k=1;
   while(k<n){
     N++;
     i=1;
     while(k<n && i<=N)
         V[++k]=i++;
   }
 }
```

a) Pt.n=10, => V={1,2,3,4,5,6,7,8,9,10}
b) Pt.n=10, => V={1,2,1,3,2,1,4,3,2,1}
c) Pt.n=12, => V={1,1,2,1,2,3,1,2,3,4}
d) Pt.n=11, => V={1,1,2,1,2,3,1,2,3,4,1}

105. Ce conţine vectorul v după execuţia codului C++:

```
 void CreV (int n, int v[]){
    int i,j,k;
    v[1]=1;
    i=2;
    k=2;
    while(i<=n){
      j=2;
      v[i++]=k;
      while(i<=n && j<=k/2){
              if(k%j==0) v[i++]=j;
              j++;
      }
      k++;
    }
}
```

a) pt n=15 => V={1,2,3,4,2,5,6,2,3,7,8,2,4,9,3}
b) pt n=15 => V={1,2,3,2,5,2,3,7,2,4,3,2,5,11,2}
c) pt n=12 => V={1,2,3,4,2,5,6,2,3,7,2,4}
d) pt n=12 => V={1,2,3,4,2,5,6,2,3,7,8,2}

106. Ce conţine vectorul Z dupa excuţîa codului C+, pentru vectorii
X={1,2,3,55,66,77,78};
Y={1,2,44,57,66,77,79,200}

```
 void A(int m, int x[], int n, int y[], int& k, int z[]){
     int i,j;
     i=j=1;
     k=0;
     while(i<=m && j<=n)
           { k++;
             if(x[i]<=y[j])  z[k]=x[i++];
             else            z[k]=y[j++];
           }
     while (i<=m) z[++k]=x[i++];
     while (j<=n) z[++k]=y[j++];
}
```

a) Z={1,1,2,3,44,55,66,77,78,79,200}
b) Z={1,1,2,2,44,55,57,66,66,77,77,78,79,200}
c) Z={1,1,2,2,3,44,55,57,66,66,77,77,78,79,200}
d) Z={1,1,2,2,3,44,55,66,77,78,79,200}

107. Ce conţine vectorul Z dupa excuţia codului C+, pentru vectorii
X={2,3,41,51,61,401};
Y={1,2,41,86,87,91,92,401}

```
 void A(int m, int x[], int n, int y[], int& k, int z[]){
     int i,j;
     i=j=1;
     k=0;
     while(i<=m && j<=n)
           { if(x[i]<=y[j]) z[++k]=x[i++];
             else           z[++k]=y[j++];
           }
     while (i<=m) z[++k]=x[i++];
     while (j<=n) z[++k]=y[j++];
  }
```

a) Z={1,3,51,62,86,87,91,92}
b) Z={1,2,3,41,51,61,86,87,91,92,401}
c) Z={1,2,3,41,41,51,61,86,87,92,401,401}
d) Z={1,2,2,3,41,41,51,61,86,87,91,92,401,401}

108. Ce conţine vectorul V după execuţia subprogramului:

```
     void creV(int n, int V[]){
           int k;
           int i;
           k=i=0;
           while(i<n){
                 k++;
                 int j=1;
                 while(i<n && j<=k)
                       V[++i]=j++;
           }

     }
```

a) n=10 => V={1,2,3,4,5,6,7,8,9,10}
b) n=12 => V={1,1,2,1,2,3,1,2,3,4}
c) n=13 => V={1,1,2,1,2,3,1,2,3,4,1,2,3}
d) n=14 => V={1,2,3,4,1,2,3,4,5,6,7,8,1,2}

109.Avem secvenţa de program:

```
 void F(int m, int p[], int n, int q[], int& k int r[]){
     k=m+n;
     for(int i=0;i<=k;i++) r[i]=0;

     for(int i=0;i<=m;i++)
       for(int j=0;j<=n;j++)
         r[i+j] += p[i] * q[j];
 }
```

Daca m=3, p={1,2,3,4}, n=2, q={1,2,3}, care propoziţii sunt adevărate ?
a) k=3 r= {10,20,30,40};
b) k=4 r= {12,13,12,10,9};
c) k=5 r= {1,4,10,16,17,12};
d) k=5 r= {1,4,10,16,18,12};

110. Un individ (mai tânăr) trebuie să urce o scară de n trepte (în drumul spre casă). Treptele le poate urca câte una sau câte două (fiind mai sportiv). În câte moduri poate urca cele n trepte?
a) n=7 => 12 moduri
b) n=8 => 21 moduri
c) n=10 => 53 moduri
d) n=11 => 144 moduri

111. O comunitate de oameni este împărţită în 3 grupuri etnice. Căte legături duale se pot forma între cele 3 grupuri etnice (legătură dual= 1 membru dintr-un grup cu un membru din alt grup)?
a) Primul grup:3 membri, al 2-lea grup:4 şi ultimul grup are 2 => 25 legături;
b) Primul grup:3 membri, al 2-lea grup:5 şi ultimul grup are 2 => 28 legături;
c) Primul grup:3 membri, al 2-lea grup 4 şi ultimul grup are 2 => 26 legături;
d) Primul grup:3 membri, al 2-lea grup 5 şi ultimul grup are 2 => 31 legături.

112. Ce returnează programul pentru n şi k numere întregi ca date de intrare.

```
int subProgram(int n,int k)
 {   int c=1;
     for(int  i=1;i<=k;i++)
        c=c*(n-i+1)/i;
     return c;
 }
```

a) n=15, k=10 => 3001;
b) n=15, k=10 => 3003;
c) n=19, k=15 => 3877;
d) n=19, k=15 => 3876;

113. Ce afişează subprogramul:

```
void pC(int a){
  int nC=log10(a);
  int p=pow(10,nC);
  int b=a;
  do {
     b=(b%10)*p+b/10;
     //cout<<b<<endl;
  }
  while (a!=b)
}
```

a) a=1234 => 2341 3412 4123 1234
b) a=1234 => 4123 1234 2341 3412
c) a=1234 => 4123 1234 3412 2341
d) a=1234 => 4123 3412 2341 1234

114. Se dau n, natural n>1, şi vectorul de întregi x. Ce valori are x după execuţia subprogramului ce urmează (alegeţi varianta corecta / variantele corecte):

```
void selectie (int n, int x[]){
   for(int i=1;i<=3;i++)
    for(int j=i+1;j<=n;j++)
      if(x[i]>x[j]) swap(x[i],x[j]);
}
```

a) n=6, x={4,5,6,3,2,1} => x ={1,2,3,4,5,6}
b) n=2, x={5,4} => x ={4,5}
c) n=5, x={5,4,3,1,2} => x ={1,2,3,4,5}
d) n=5, x={5,4,3,1,2} => x ={1,2,3,5,4}

115. Ce valori au varibilele ls, ld, la terminarea instrucţiunii repetitive sau inainte de instructiunea return m?

```
int cautBinar(int n, int x[], int val){
    int ls,ld,m;  //n lung lui x
    ls=1;
    ld=n;
    do
      { m=(ls+ld)/2;
        if(x[m]==val) return m;
        if(val<x[m]) ld=m-1;
        else         ls=m+1;
      }
    while (ls<=ld);
    return -1;
}
```

a) n=6, x={1,2,3,4,5,6} val = 10 => ls=7, ld=8
b) n=6, x={1,2,3,4,5,6} val = 10 => ls=7, ld=5
c) n=6, x={1,2,3,4,5,6} val = 4 => ls=5, ld=6
d) n=6, x={1,2,3,4,5,6} val = 4 => ls=4, ld=4