Examen de licenţă 2018 - Informatică
Exemple de întrebări - Structuri discrete şi algoritmi

In atenţia studenţilor:

Proba scrisă a examenului de licenţă din sesiunile iulie-septembrie 2018 va consta din 60 de întrebări similare, ca structură şi nivel de dificultate, celor din această culegere. Pentru fiecare dintre cele trei categorii (Structuri discrete şi algoritmi, Limbaje de programare şi inginerie software, Sisteme de calcul şi baze de date) vor fi câte 20 de întrebări.

Pentru neclarităţi privind enunţurile sau răspunsurile puteţi să vă adresaţi celor care au propus întrebările pentru fiecare secţiune.

Algoritmi:

- Daniela Zaharie (daniela.zaharie@e-uvt.ro)

Logica computaţională:

- Adrian Crăciun (adrian.craciun@e-uvt.ro)

Structuri de date:

- Cosmin Bonchiş (cosmin.bonchis@e-uvt.ro)

Teoria grafurilor şi combinatorică:

- Mircea Marin (mircea.marin@e-uvt.ro)

Limbaje formale şi teoria automatelor:

- Mircea Drăgan (mircea.dragan@e-uvt.ro)

1 Algoritmi

1. Se consideră secvenţa:

```
a:=1
WHILE a<>n DO
  a=2*a
ENDWHILE
```

Pentru ce valori ale lui n ciclul de mai sus NU este infinit?

(a) pentru orice valoare naturală nenulă
(b) pentru orice valoare naturală nenulă pară
(c) pentru orice putere a lui 2
(d) pentru orice putere pară a lui 2
(e) pentru orice putere impară a lui 2

2. Se consideră secvenţa:

```
a:=0
FOR i:=1,n DO
  i:=i-h
  a:=a+1
ENDFOR
```

Pentru ce valori ale variabilei întregi h ciclul de mai sus este infinit?

(a) pentru `h`=0
(b) pentru orice `h` negativ
(c) pentru nici o valoare a lui `h`
(d) pentru orice `h` mai mare sau egal cu 1

3. Se consideră secvenţa:

```
c:=a MOD q
a:=a DIV q
WHILE a>0 DO
   IF c<a MOD q THEN c:=a MOD q ENDIF
   a:=a DIV q
ENDWHILE
```

Ştiind că variabila `a` conţine o valoare naturală, iar `q` este o valoare naturală cuprinsă între 2 şi 10, după execuţia secvenţei, variabila c va conţine:

(a) cifra de pe poziţia cea mai puţin semnificativă din reprezentarea în baza q a numărului `a`

(b) cifra de pe poziţia cea mai semnificativă din reprezentarea în baza `q` a numărului `a`

(c) cifra minimă întâlnită în reprezentarea în baza `q` a numărului `a`

(d) cifra maximă întâlnită în reprezentarea în baza `q` a numărului `a`

(e) cel mai mare multiplu al lui `q` care îl divide pe `a`

4. Se consideră un tablou `b[0..n]` care conţine cifrele reprezentării binare a unui număr natural (`b[n]` corespunde celei mai semnificative cifre, iar `b[0]` celei mai puţin semnificative cifre) şi care are proprietatea că există cel puţin un indice `i` pentru care `b[i]=0`. Stabiliţi care dintre afirmaţiile de mai jos sunt adevărate pentru algoritmul următor:

```
alg(b[0..n])
  i:=0
  WHILE b[i]=1 DO
    b[i]:=0
    i:=i+1
  ENDWHILE
  b[i]:=1
  RETURN b[0..n]
```

(a) Transformă toate elementele din `b[0..n]` care sunt egale cu 1 în 0

(b) Aparţine clasei de complexitate $\Theta(n)$

(c) Aparţine clasei de complexitate $\mathcal{O}(n)$

(d) Realizează incrementarea cu 1 a valorii naturale reprezentate în baza doi prin tabloul `b[0..n]`

5. Se consideră un tablou `b[0..n]` care conţine cifrele reprezentării binare a unui număr natural (`b[n]` corespunde celei mai semnificative cifre, iar `b[0]` celei mai puţin semnificative cifre) şi care are proprietatea că există cel puţin un indice `i` pentru care `b[i]=1`. Stabiliţi care dintre afirmaţiile de mai jos sunt adevărate pentru algoritmul următor:

```
alg(b[0..n])
  i:=0
  WHILE b[i]=0 DO
    b[i]:=1
    i:=i+1
  ENDWHILE
  b[i]:=0
  RETURN b[0..n]
```

(a) Aparţine clasei de complexitate $\mathcal{O}(n)$

(b) Aparţine clasei de complexitate $\Theta(n)$

(c) Realizează decrementarea cu 1 a valorii naturale reprezentate în baza doi prin tabloul `b[0..n]`

(d) Transformă toate elementele din `b[0..n]` care sunt egale cu 0 în 1

6. Se consideră secvenţa:

```
d:=m
i:=n
r:=d MOD i
WHILE r<>0 DO
  d:=i
  i:=r
  r:=d MOD i
ENDWHILE
```

Pentru ce valori ale lui `m` şi `n`, variabila `i` va conţine după execuţia prelucrărilor cel mai mare divizor comun al lui m şi n?

(a) pentru `m` şi `n` naturale nenule ce satisfac `m<n`

(b) pentru `m` şi `n` naturale nenule ce satisfac `m>n`

(c) pentru orice numere naturale nenule `m` şi `n`

(d) pentru nici o valoare

7. Se consideră secvenţa:

```
FOR d:=2,n DO
  IF n MOD d=0 THEN
    WRITE d
    WHILE (n MOD d=0) DO
      n:=n DIV d
    ENDWHILE
  ENDIF
ENDFOR
```

Pentru un număr natural `n`, prin execuţia secvenţei se vor afişa:

(a) toţi divizorii pozitivi ai lui `n`

(b) toţi divizorii lui `n`

(c) toţi divizorii naturali ai lui `n` care sunt numere prime

(d) toţi divizorii naturali ai lui `n` care sunt numere impare

(e) nici o valoare

8. Fie `x[1..n]` un tablou cu elemente întregi şi secvenţa:

```
FOR i:=1,m DO
  a:=x[i]
  x[i]:=x[n-i+1]
  x[n-i+1]:=a
ENDFOR
```

Ce valoare ar trebui să aibă `m` astfel încât efectul secvenţei de mai sus să fie inversarea ordinii elementelor tabloului şi numărul de interschimbări să fie cât mai mic?

(a) `n`

(b) `n DIV 2`

(c) `n DIV 2 +1`

(d) `3n DIV 2`

(e) nici o valoare a lui `m` nu asigură inversarea ordinii elementelor tabloului

9. Se consideră un tablou `x[1..n]` şi secvenţa de prelucrări:

```
k1:=1
k2:=1
FOR i:=2,n DO
    IF x[k1]>x[i] THEN k1:=i
                  ELSE IF x[k2]<=x[i] THEN k2:=i ENDIF
    ENDIF
ENDFOR
```

Care dintre următoarele afirmaţii sunt adevărate după execuţia prelucrărilor?

(a) `x[k1]`$\leq$`x[k2]`

(b) `x[k1]`$\geq$`x[k2]`

(c) `k1` este ultima poziţie pe care se află valoarea minimă din tablou, iar `k2` este prima poziţie pe care se află valoarea maximă din tablou

(d) `k1` este prima poziţie pe care se află valoarea minimă din tablou, iar `k2` este ultima poziţie pe care se află valoarea maximă din tablou

(e) `k1` este prima poziţie pe care se află valoarea maximă din tablou, iar `k2` este ultima poziţie pe care se află valoarea minimă din tablou

(f) `k1` este ultima poziţie pe care se află valoarea maximă din tablou, iar `k2` este prima poziţie pe care se află valoarea minimă din tablou

10. Fie `x[1..n]` un tablou cu elemente reale. Se consideră secvenţa de prelucrări:

```
i:=1
j:=n
WHILE i<j DO
  WHILE (x[i]<0) AND (i<n) DO i:=i+1 ENDWHILE
  WHILE (x[j]>0) AND (j>1) DO j:=j-1 ENDWHILE
  IF i<j THEN
    aux:=x[i]
    x[i]:=x[j]
    x[j]:=aux
  ENDIF
ENDWHILE
```

Ce efect are secvenţa de prelucrări asupra tabloului `x` ?

(a) ordonează crescător elementele lui `x`

(b) elimină toate valorile pozitive din `x`

(c) elimină toate valorile negative din `x`

(d) plasează toate elementele negative înaintea elementelor pozitive;

(e) plasează toate elementele pozitive înaintea celor negative

11. Se consideră o matrice pătratică, `a[1..n,1..n]`, cu elemente din $\{0, 1\}$ şi algoritmul:

```
alg (a[1..n,1..n])
FOR r:=1,n-1 DO
  s1:=0; s2:=0;
  FOR i:= r+1,n DO
    s1:=s1+a[i,i-r]
    s2:=s2+a[i-r,i]
  ENDFOR
  IF NOT(s1=1) OR NOT(s2=1) THEN RETURN False ENDIF
ENDFOR
RETURN True
```

Care dintre următoarele afirmaţii este(sunt) adevarată(e) pentru algoritmul de mai sus:

(a) Algoritmul returnează True în toate cazurile în care fiecare linie şi fiecare coloană a matricii conţine o singură valoare egală cu 1

(b) Algoritmul returnează True doar dacă fiecare dintre diagonalele paralele cu diagonala principală conţine exact un element egal cu 1

(c) Algoritmul execută de n(n+1)/2 ori corpul ciclului FOR după `i`

(d) Algoritmul returnează False doar dacă toate diagonalele paralele cu diagonala principalã a matricii conţin doar elemente egale cu 0

(e) Algoritmul execută de n(n-1)/2 ori corpul ciclului FOR după `i`

12. Se consideră algoritmul:

```
FOR k:=0,n-1 DO
  FOR j:=1,n-k DO
     WRITE a[j+k,j]
  ENDFOR
ENDFOR
```

Numărul de afişări efectuate este:

(a) $n^2/2$

(b) $n + n^2/2$

(c) $n(n-k)$

(d) $n(n+1)/2$

13. Se consideră algoritmul:

```
FOR i:=1,n-1 DO
   IF x[i]>x[i+1] THEN
     aux:=x[i]
     x[i]:=x[i+1]
     x[i+1]:=aux
   ENDIF
ENDFOR
```

Ordinul de complexitate al algoritmului în raport cu numărul de interschimbări efectuate este:

(a) $\mathcal{O}(\lg n)$

(b) $\mathcal{O}(n)$

(c) $\mathcal{O}(n \lg n)$

(d) $\mathcal{O}(n^2)$

(e) $\Theta(n)$

14. Fie `x[1..n]` un tablou cu elemente reale ordonate crescător ($n > 1$), iar `v` o valoare reală oarecare. Se consideră algoritmul:

```
s:=1
d:=n
r:=0
WHILE (s<=d) AND (r=0) DO
    m:=(s+d) DIV 2
    IF v=x[m] THEN r:=1
              ELSE IF v<x[m] THEN d:=m-1
                             ELSE s:=m+1
                   ENDIF
    ENDIF
ENDWHILE
```

Care dintre următoarele afirmaţii referitoare la algoritmul de mai sus sunt adevărate?

(a) numărul minim de comparaţii cu valoarea `v` este 2

(b) ordinul de complexitate în raport cu numărul de comparaţii este $\mathcal{O}(n)$

(c) ordinul de complexitate în raport cu numărul de comparaţii este $\mathcal{O}(\lg n)$

(d) numărul minim de comparaţii cu valoarea `v` este 1

(e) variabila `r` conţine valoarea 1, dacă `v` se află în tablou şi 0 în caz contrar

(f) variabila `r` conţine valoarea 0, dacă `v` se află în tablou şi 1 în caz contrar

15. Fie `x[1..n]` un tablou. Se consideră următorul algoritm:

```
nr:=0
i:=1
WHILE i<=n DO
  k:=0
  WHILE (x[i+k]>0) AND ((i+k)<n) DO k:=k+1 ENDWHILE
  IF nr<k THEN nr:=k ENDIF
  i:=i+k+1
ENDWHILE
```

Care este ordinul de complexitate al algoritmului?

(a) $\mathcal{O}(n^2)$

(b) $\mathcal{O}(n)$

(c) $\mathcal{O}(\lg n)$

(d) $\mathcal{O}(n \cdot k)$

(e) $\mathcal{O}(n \cdot \lg k)$

16. Fie `x[1..n]` un tablou neordonat. Se pune problema determinării unei perechi, $(imax, jmax)$, de indici din $\{1, 2, \ldots, n\}$ care are proprietatea că $|x[imax] - x[jmax]|$ este maximă. Care dintre următorii algoritmi realizează acest lucru în $\mathcal{O}(n)$?

(a)
```
imax:=1; jmax:=2
FOR i:=1,n-1 DO
  FOR j:=i+1,n DO
    IF |x[i]-x[j]|>|x[imax]-x[jmax]| THEN imax:=i; jmax:=j ENDIF
  ENDFOR
ENDFOR
```

(b)
```
imax:=1; jmax:=1
FOR i:=2,n DO
  IF x[imax]>x[i] THEN imax:=i ENDIF
  IF x[jmax]<x[i] THEN jmax:=i ENDIF
ENDFOR
```

(c)
```
imax:=1; jmax:=1
FOR i:=2,n DO
  IF x[imax]<x[i] THEN imax:=i ENDIF
  IF x[jmax]>x[i] THEN jmax:=i ENDIF
ENDFOR
```

17. Se consideră un tablou, `a[1..n]`, ordonat crescător şi o valoare `x`. Se pune problema verificării existenţei cel a puţin unei perechi de elemente `a[i]` şi `a[j]` ($i \neq j$)care satisfac proprietatea `a[i]+a[j]=x` şi se consideră următorii algoritmi:

```
alg1 (a[1..n],x)
  i:=1; j:=n
  WHILE i<j DO
    IF a[i]=x-a[j] THEN RETURN True
       ELSE IF a[i]<x-a[j] THEN i:=i+1 ELSE j:=j-1 ENDIF
    ENDIF
  ENDWHILE
  RETURN False
```

```
alg2(a[1..n],x)
  FOR i:=1,n DO
    FOR j:=1,n DO
      IF a[i]+a[j]=x THEN RETURN True
    ENDFOR
  ENDFOR
  RETURN False
```

```
alg3(a[1..n],x)
  FOR i:=1,n-1 DO
    FOR j:=i+1,n DO
      IF a[i]+a[j]=x THEN RETURN True
```

```
        ELSE RETURN FALSE
      ENDFOR
    ENDFOR
```

Care dintre următoarele afirmaţii este(sunt) adevarată(e)?

(a) Algoritmul alg1 rezolvă corect problema şi are ordinul de complexitate $\mathcal{O}(n)$

(b) Algoritmul alg3 are un ordin de complexitate mai mic decăt algoritmul alg2

(c) Algoritmul alg2 rezolvă corect problema şi are ordinul de complexitate $\mathcal{O}(n^2)$

(d) Algoritmul alg3 rezolvă corect problema şi are ordinul de complexitate $\mathcal{O}(n^2)$

18. Se consideră secvenţa de prelucrări:

```
k:=0
i:=1
WHILE i<=n DO
   k:=k+1
   i:=2*i
ENDWHILE
```

Numărul de înmulţiri efectuate în secvenţa de mai sus este de ordinul:

(a) $\mathcal{O}(n)$

(b) $\mathcal{O}(n^2)$

(c) $\mathcal{O}(\lg n)$

(d) $\mathcal{O}(2n)$

(e) $\mathcal{O}(n/2)$

19. Se consideră secvenţa de prelucrări:

```
s:=0
m:=1
FOR i:=1,n DO
   m:=2*m
   FOR j:=1,m DO s:=s+1 ENDFOR
ENDFOR
```

Numărul de incrementări ale variabilei `s` este de ordinul:

(a) $\mathcal{O}(n)$

(b) $\mathcal{O}(n^2)$

(c) $\mathcal{O}(n \lg n)$

(d) $\mathcal{O}(2^n)$

(e) $\mathcal{O}(n^4)$

20. Se consideră următorii algoritmi pentru calculul factorialului unui număr natural:

```
fact1(n)
  f:=1
  FOR i:=2,n DO
    f:=f*i
  ENDFOR
  RETURN f

fact2(n)
IF n<=1 THEN RETURN 1
     ELSE RETURN n*fact2(n-1)
  ENDIF
```

Care dintre următoarele afirmaţii este(sunt) adevărată(e) în condiţiile în care în analiza complexităţii se ia în considerare doar operaţia de înmulţire?

(a) ordinul de complexitate al algoritmului `fact1` este mai mare decât cel al algoritmului `fact2`

(b) ordinul de complexitate al algoritmului `fact2` este mai mare decât cel al algoritmului `fact1`

(c) algoritmii `fact1` şi `fact2` au acelaşi ordin de complexitate

21. Se consideră algoritmul:

```
alg(n)
  IF n>0 THEN
    alg(n DIV 2)
    WRITE n MOD 2
  ENDIF
```

Dacă este apelat pentru un număr natural nenul, `n`, algoritmul va afişa:

(a) cifrele corespunzătoare reprezentării în baza 2 a numărului `n` (începând de la cifra cea mai puţin semnificativă)

(b) cifrele corespunzătoare reprezentării în baza 2 a numărului `n` (începând de la cifra cea mai semnificativă)

(c) restul împărţirii lui `n` la 2

(d) câtul împărţirii lui `n` la 2

22. Se consideră algoritmul:

```
alg(n)
  IF n=0 THEN RETURN 0
         ELSE RETURN n MOD 2+alg(n DIV 2)
  ENDIF
```

Presupunând că algoritmul este apelat pentru un număr natural, `n`, care dintre afirmaţiile de mai jos sunt adevărate?

(a) algoritmul returnează numărul de cifre din reprezentarea binară a lui `n`

(b) algoritmul returnează numărul de cifre egale cu 1 din reprezentarea binară a lui `n`

(c) algoritmul returnează suma tuturor cifrelor din reprezentarea binară a lui `n`

(d) algoritmul returnează numărul de cifre egale cu 0 din reprezentarea binară a lui `n`

23. Se consideră următorul algoritm recursiv care are acces la conţinutul unui tablou x[1..n]:

```
alg(i, j)
  IF i<j THEN
    aux:=x[i]
    x[i]:=x[j]
    x[j]:=aux
    alg(i+1,j-1)
  ENDIF
```

Efectul apelului `alg(1,n)` este:

(a) interschimbă primul element cu ultimul element al tabloului `x[1..n]`

(b) inversează ordinea tuturor elementelor tabloului `x[1..n]`

(c) lasă tabloul `x` nemodificat

(d) interschimbă elementele vecine din tabloul iniţial

24. Se consideră un tablou `x[1..n]` şi algoritmul:

```
sort(x[1..n])
 FOR i:=2,n DO
   aux:=x[i]
   j:=i-1
   WHILE (j>=1) AND <conditie> DO
      x[j+1]:=x[j]
      j:=j-1
   ENDWHILE
   x[j+1]:=aux
 ENDFOR
```

Cu care dintre expresiile de mai jos trebuie completată condiţia de la `WHILE` astfel încât algoritmul să realizeze ordonarea crescătoare a tabloului `x`?

(a) `x[j]<=aux`

(b) `x[j]>=aux`

(c) `x[j]<aux`

(d) `x[j]>aux`

25. Se consideră un tablou ordonat crescător, `x[1..n]`, o valoare `v` şi algoritmul:

```
alg(x[1..n],v)
  g:=0
  i:=1
  WHILE (g=0) AND (i<=n) DO
    IF v<=x[i] THEN g:=1 ELSE i:=i+1 ENDIF
  ENDWHILE
RETURN i
```

Care dintre următoarele afirmaţii sunt adevărate?

(a) algoritmul are complexitate liniară

(b) la ieşirea din ciclu, `g` are valoarea 0 dacă şi numai dacă `v>x[n]`

(c) algoritmul returnează numărul de elemente din `x` care sunt strict mai mici decât `v`

(d) algoritmul returnează poziţia pe care poate fi inserat `v` astfel încât tabloul să rămână ordonat crescător

(e) algoritmul returnează întotdeauna (`n`+1)

26. Se consideră un tablou `x[1..n]` şi algoritmul:

```
Alg(x[1..n])
  FOR i:=n,2,-1 DO
    FOR j:=1,i-1  DO
      IF x[j]<x[j+1] THEN
         aux:=x[j]
         x[j]:=x[j+1]
         x[j+1]:=aux
      ENDIF
    ENDFOR
  ENDFOR
  RETURN x[1..n]
```

Care dintre următoarele afirmaţii sunt adevărate?

(a) algoritmul ordonează crescător tabloul `x[1..n]`

(b) algoritmul ordonează descrescător tabloul `x[1..n]`

(c) algoritmul nu realizează nici ordonarea crescătoare nici cea descrescătoare a tabloului;

(d) indiferent de configuraţia iniţială a tabloului algoritmul are complexitate liniară (aparţine lui $\Theta(n)$);

(e) indiferent de configuraţia iniţială a tabloului algoritmul are complexitate pătratică (aparţine lui $\Theta(n^2)$)

27. Se consideră o variabilă reală `x`, o variabilă naturală nenulă n şi algoritmul:

```
alg(x,n)
  IF n=1 THEN RETURN x
         ELSE
           p:=alg(x,n DIV 2)
           IF n MOD 2=0 THEN RETURN p*p
                        ELSE RETURN p*p*x
           ENDIF
  ENDIF
```

Care dintre următoarele afirmaţii sunt adevărate?

(a) algoritmul returnează x^2 dacă `n` este par şi x^3 dacă `n` este impar

(b) este posibil ca succesiunea de apeluri recursive să nu se termine

(c) algoritmul returnează x^n dacă `n` este par x^{n+1} şi dacă `n` este impar

(d) algoritmul returnează x^n indiferent de paritatea lui n

(e) algoritmul are complexitate liniară ($\mathcal{O}(n)$)

(f) algoritmul are complexitate logaritmică ($\mathcal{O}(\lg n)$)

28. Se consideră două valori naturale `a` şi `b` şi algoritmul:

```
alg(a,b)
 IF a=0 OR b=0 THEN RETURN 0
 ELSE
    IF a MOD 2=0 THEN  RETURN (alg(a DIV 2,2*b))
      ELSE RETURN (alg(a DIV 2,2*b)+b)
    ENDIF
 ENDIF
```

Care dintre următoarele afirmaţii sunt adevărate?

(a) este posibil ca succesiunea de apeluri recursive să nu se termine

(b) algoritmul returnează întotdeauna 0

(c) algoritmul returnează produsul `a*b` dacă `a` este par şi `a*b+b` dacă `a` este impar

(d) algoritmul returnează produsul `a*b` indiferent de paritatea lui `a`

29. Se consideră algoritmul `alg` apelat pentru un număr natural `n` şi se notează cu $T(n)$ numărul de operaţii de adunare efectuate:

```
alg(n)
  IF n<=9 THEN RETURN n
          ELSE RETURN alg(n DIV 10)+n MOD 10
ENDIF
```

Care dintre următoarele afirmaţii este(sunt) adevarată(e):

(a) Algoritmul returnează câte cifre nenule are numărul n

(b) Algoritmul returnează suma cifrelor numărului `n`

(c) $T(n) = 1$ dacă $n \leq 9$ şi $T(n) = T(nDIV10) + nMOD10$ dacă $n > 9$

(d) $T(n) = 0$ dacă $n \leq 9$ şi $T(n) = T([n/10]) + 1$ dacă $n > 9$

(e) T(n) aparţine lui $\mathcal{O}(n)$

(f) T(n) aparţine lui $\mathcal{O}(\lg n)$

30. Se consideră că timpul de execuţie al unui algoritm recursiv folosit pentru rezolvarea unei probleme de dimensiune `n` satisface următoarea relaţie de recurenţă:

$T(n) = 1$ dacă $n = 1$, respectiv $T(n) = nT(n-1) + 2$ dacă $n > 1$

Stabiliţi cărei clase de complexitate aparţine $T(n)$:

(a) $\mathcal{O}(2^n)$

(b) $\mathcal{O}(n!)$

(c) $\mathcal{O}(n^2)$

(d) $\mathcal{O}(2n)$

2 Logică computaţională

1. Consideraţi, în logica predicatelor, limbajul care conţine următoarele simboluri:

   - variabile, pentru care se folosesc litere mici,
   - simboluri funcţionale $\mathcal{F}$: $+$ binar, infix, $-$ unar, prefix, $*$ binar, infix.
   - simboluri predicative $\mathcal{P}$: $=, <, \leq$ toate binare, infix.
   - simboluri pentru constante $\mathcal{C}$: $0, 1$.

   Care din următoarele sunt termeni peste acest limbaj?

   (a) $(0 * x) - 1$,
   (b) $1 + (z * x) < 0$,
   (c) $x + ((-1) * 0)$,
   (d) $0 * (y + 1)$.

2. Consideraţi, în logica predicatelor, limbajul care conţine următoarele simboluri:

   - variabile, pentru care se folosesc litere mici,
   - simboluri funcţionale $\mathcal{F}$: $+$ binar infix, $-$ unar prefix, $*$ binar infix.
   - simboluri predicative $\mathcal{P}$: $=, <, \leq$ toate binare, infix.
   - simboluri pentru constante $\mathcal{C}$: $0, 1$.

   Care din următoarele sunt formule peste acest limbaj?

   (a) $\forall x \forall y((x \leq 1) \Rightarrow (1 = 0))$,
   (b) $(x = y) \wedge (1 + 0)$,
   (c) $(x < (1 + 0)) \vee \forall x \exists y(x = (-y))$,
   (d) $\forall x((x - 1) = y) \wedge (x < 0)$.

3. Care din următoarele sunt domenii inductive?

   (a) numerele naturale,
   (b) numerele întregi,
   (c) formulele logicii propoziţiilor,
   (d) liste de numere întregi.

4. Folosind definiţia formulelor propoziţionale bine formate, decideţi care din următoarele sunt formule propoziţionale:

   (a) $(((P \rightarrow Q) \vee S) \leftrightarrow T)$,
   (b) $(P \rightarrow (Q \wedge (S \rightarrow T)))$,

(c) $(\neg(B(\neg Q)) \wedge R)$.

5. Pentru următoarele formule propoziţionale, şi pentru interpretarea $\{P, \neg Q\}$:

   (a) $((P \to Q) \wedge ((\neg Q) \wedge P))$ are valoarea sub interpretare $\mathbb{A}$,

   (b) $((P \to Q) \to (Q \to P)$ are valoarea sub interpretare $\mathbb{A}$,

   (c) $((\neg(P \vee Q)) \wedge (\neg Q))$ are valoarea sub interpretare $\mathbb{F}$.

6. Care din următoarele afirmaţii este adevărată:

   (a) dacă o formulă propoziţională este validă, atunci este satisfiabilă,

   (b) dacă o formulă propoziţională nu este validă, atunci este nesatisfiabilă,

   (c) dacă o formulă propoziţională nu este validă, atunci negaţia sa este satisfiabilă,

   (d) dacă o formulă propoziţională nu este validă, atunci negaţia sa este validă.

7. Care este relaţia dintre propoziţiile

$$(F \wedge G) \to H$$

   şi

$$F \to (G \to H)?$$

   (a) sunt logic echivalente

   (b) prima este o consecinţă logică a celei de-a doua,

   (c) a doua este o consecinţă logică a primeia,

   (d) nu se relaţionează în niciun fel descris mai sus.

8. Formula

$$P \leftrightarrow Q$$

   este

   (a) logic echivalentă cu conjuncţia formulelor de mai jos,

   (b) o consecinţă logică a formulelor de mai jos,

   (c) logic echivalentă cu disjuncţia forulelor de mai jos,

   aceste formule fiind:

$$Q \to R,$$
$$R \to (P \wedge Q),$$
$$P \to (Q \vee R).$$

9. Care din formulele de mai jos sunt în formă normală disjunctivă?

   (a) P,

   (b) $\neg P \vee Q$,

(c) $P \wedge \neg Q \wedge S$,

(d) $(P \wedge \neg Q \wedge S) \vee \neg S$.

10. Care din următoarele formule reprezintă o formă normală disjunctivă a formulei

$$(P \to Q)?$$

(a) $\top$,

(b) $\bot$,

(c) $\neg P \vee Q$,

(d) $(\neg P \wedge \neg Q) \vee (\neg P \wedge Q) \vee (P \wedge Q)$.

11. Care din următoarele este un rezolvent al clauzelor $\{P, \neg Q, R\}$ şi $\{\neg P, Q, S\}$?

(a) $\emptyset$,

(b) $\{P, \neg P, R, S\}$,

(c) $\{R, S\}$.

12. Pentru a verifica faptul că o formulă G este o consecinţă logică a formulelor $F_1, \ldots, F_n$, se poate:

(a) verifica dacă $(F_1 \wedge \ldots \wedge F_n) \to G$ este nesatisfiabilă,

(b) verifica dacă $\neg F_1 \vee \ldots \vee \neg F_n \vee G$ este nesatisfiabilă,

(c) verifica dacă $\neg F_1 \vee \ldots \vee \neg F_n \vee G$ este validă,

(d) verifica dacă $F_1 \wedge \ldots \wedge F_n \wedge \neg G$ este nesatisfiabilă.

13. Există o formulă logic echivalentă cu

$$\left(\begin{array}{c} ((P_1 \to (P_2 \vee P_3) \wedge (\neg P_1 \to (P_3 \vee P_4)))) \\ \wedge \\ ((P_3 \to (\neg P_6)) \wedge (\neg P_3 \to (P_4 \to P_1))) \\ \wedge \\ (\neg(P_2 \wedge P_5)) \wedge (P_2 \to P_5) \end{array} \right) \to \neg(P_3 \to P_6),$$

care conţine doar conectori propoziţionali din mulţimea:

(a) $\{\neg, \vee\}$,

(b) $\{\vee, \wedge\}$,

(c) $\{|\}$,

(d) $\{\bot, \to\}$.

unde $|$ este conectorul NAND (i.e. $P|Q = \neg(P \wedge Q)$).

14. Care dintre următoarelee metode din logica propoziţională sunt metode de raţionament (verifică proprietăţile raţionamentului)?

(a) tabelele de adevăr,

(b) rezoluţia,

(c) metoda Davis Putnam.

15. Consideraţi funcţia booleană cu trei argumente ce implementează paritatea: returnează $\mathbb{A}$ dacă numărul de argumente cu valoarea $\mathbb{A}$ este impar, $\mathbb{F}$ altfel. Care dintre următoarele formule descrie această funcţie?

(a) $(P \wedge Q \wedge R) \rightarrow (P \vee Q \vee R)$,

(b) $(\neg P \wedge \neg Q \wedge R) \vee (\neg P \wedge Q \wedge \neg R) \vee (P \wedge \neg Q \wedge \neg R) \vee (P \wedge Q \wedge R)$,

(c) $(\neg P \rightarrow (Q \wedge R)) \vee (\neg R \rightarrow (P \wedge Q)) \vee (\neg Q \rightarrow (P \wedge R))$.

16. Mulţimea de clauze ce corespunde formulei

$$(\neg P \rightarrow (Q \wedge R)) \rightarrow (P \rightarrow \neg Q)$$

este:

(a) $\{\{\neg P, \neg Q\}\}$,

(b) $\{\{P, \neg Q\}, \{P, R\}, \{\neg Q, R\}\}$,

(c) $\{\{P, \neg Q, \neg R\}, \{P, Q, R\}, \{\neg P, \neg Q, R\}\}$.

17. Consideraţi mulţimea de clauze:

(1) $\{P, Q, \neg R\}$,
(2) $\{\neg P, R\}$,
(3) $\{P, \neg Q, S\}$,
(4) $\{\neg P, \neg Q, \neg R\}$,
(5) $\{P, \neg S\}$.

Formula corespunzătoare acestei mulţimi este:

(a) validă,

(b) satisfiabilă,

(c) nesatisfiabilă.

18. Metoda Davis-Putnam returnează răspunsul satisfiabil:

(a) când se generează clauza vidă,

(b) când se generează mulţimea vidă de clauze,

(c) când nu se pot genera clauze noi, şi clauza vidă nu este în mulţimea de clauze.

19. Consideraţi mulţimea de clauze:

$$\{\{F, \neg G, H\}, \{\neg F, \neg G\}, \{G, H\}, \{\neg F, H\}, \{F, G, \neg H\}\}.$$

Care dintre regulile metodei Davis-Putnam-Logemann-Loveland se poate aplica în acest caz?

(a) regula literalului pur,

(b) regula clauzei cu 1 literal,

(c) regula de împărţire (splitting).

20. Pentru a demonstra o formulă G când se cunoaşte o disjuncţie $A \vee B$:

(a) se presupune A şi se demonstrează G, apoi se presupune B şi se demonstrază G,

(b) se presupune A şi se demonstrează G,

(c) se presupune $\neg A$ şi se demonstrează B şi G.

3 Structuri de date

/ * Se presupune, acolo unde este cazul, că fişierele care conţin prototipul funcţiilor apelate din biblioteca C, vor fi corect incluse în programele care apelează funcţiile din testele enunţate */

1. Componentele unei structuri de date:
   (a) trebuie să fie toate de acelaşi tip;
   (b) pot fi la rândul lor structuri de date;
   (c) pot fi toate de acelasi tip;
   (d) nu pot fi tipuri definite de către utilizator;
   (e) pot avea unul din tipurile standard ale limbajului de programare utilizat.
2. Se numeşte listă liniară:
   (a) structura de date în care inserările, suprimările şi orice alt acces se efectuează la unul din capetele listei;
   (b) o colecţie de $n \geq 0$ elemente de acelaşi tip, ale căror proprietăţi structurale se reduc la poziţiile relative liniare (unidimensionale) ale elementelor în cadrul structurii;
   (c) o structură de date asimilată cu un tablou in care elementele sunt memorate într-o zonă continuă de memorie, in locaţii succesive;
   (d) o structură avansată de date ce se caracterizează printr-un grad de abstractizare ridicat.
3. Selectaţi care din următoarele operaţii sunt considerate operaţii de bază asupra listelor în general:
   (a) Sortarea elementelor listei în ordinea crescătoare/ descrescătoare a valorilor anumitor câmpuri;
   (b) Insumarea elementelor listei ;
   (c) Accesul la elementul de pe poziţia i din listă;
   (d) Suprimarea unui nod din listă ;
   (e) Copierea unei liste pe un alt suport;
4. Care din declaraţiile de mai jos definesc corect o structură de tip listă liniară l, implementată prin tipul tablou?
   (a)
```
#define lungmax 5
typedef int nod;
typedef struct{
nod elemente[lungmax];
int ultim;
}lista;
lista l;
```

(b)
```
#define lung_max 5
    struct {
        nod elemente [lung_max];
        int ultim;
    }l;
```

(c)
```
#define lung_max 5
    typedef int nod;
    typedef struct {
        nod elemente [lung_max];
    }l;
```

(d)
```
#define lung_max 5
    typedef int nod;
    struct {
        nod elemente [lung_max];
        int ultim;
    } ;
    lista l;
```

5. Se consideră descrierile în limbajul C:

```
#include <stdio.h>
#include <conio.h>
#define lung_max 5
typedef int nod;

typedef struct {
nod elemente[lung_max];
    int ultim; / * indexul ultimului element al listei */

}lista;
lista l;
int k; / * k indica o pozitie din interiorul listei*/
void initializare(void) {
...
l.ultim=-1;
} / * initializare */

void ins(void) {
    int i;
    if(l.ultim>=lungime_max-1)
        printf(  "Depasire\n  ");
```

```
    else {
        for(i=l.ultim;i>=k;i--)
            l.elemente[i+1]=l.elemente[i];
        printf(  "Introduceti val. noului el.(nr.intreg):  ");
        scanf(  "%d  ",&l.elemente[k]);
        printf(  "\n  ");

        l.ultim++;
    }
} /* insp */
```

şi o listă l astfel implementată, cu *l.elemente=(3, 2, 5, 7,0), l.ultim=3.* Care va fi imaginea listei l după apelul funcţiei *ins*,

pentru k=2 şi citirea de la tastatură a numărului 9 ?

(a) `l.element = {3, 9, 2, 5, 7} l.ultim = 4;`

(b) `l.element = {3, 2, 9, 5, 7} l.ultim = 4;`

(c) `l.element = {2, 5, 5, 7, 3} l.ultim = 4;`

(d) `l.element = {3, 2, 5, 7, 9} l.ultim = 4.`

6. Se consideră descrierile în limbajul C:

```
#include <stdio.h>
#include <conio.h>
#define lung_max 20
typedef int nod;

typedef struct {
 nod elemente[lung_max];
 int ultim;
}lista;
lista l;
int i;
/ * i indica o pozitie din interiorul listei*/
void initializare(void) {
l.ultim=-1;
} / * initializare */
```

Ce înţelegeţi prin suprimarea nodului de pe poziţia i, diferit de ultimul nod, dintr-o listă astfel implementată?

(a) Se atribuie valoarea 0 elementului de pe pozitia i;

(b) Se decrementeaza câmpul ultim, apoi se deplaseaza toate elementele de după poziţia i spre sfârşitul tabloului;

(c) Se deplasează toate elementele de după poziţia i cu o poziţie spre începutul tabloului, apoi se decrementează valoarea campului ultim;

(d) Se deplasează toate elementele de dinaintea poziţiei i spre începutul tabloului, apoi se decrementează valoarea campului ultim.

7. O listă liniară simplu înlănţuită este:

(a) lista liniară, în care relaţia de ordonare este materializată pe suport printr-n pointer către elementul următor;

(b) o structură de date implementată prin tipul tablou;

(c) o structură explicită si recursivă;

8. Se consideră descrierile în limbajul C:

```
struct nod {
int cheie;
char info[10];
struct nod *urm;
};
typedef struct nod Tnod;

typedef Tnod *ref;
ref p=NULL,q;
/* p - pointerul spre primul element al listei */
```

Se dă funcţia:

```
void func(void) {
q=(ref)malloc(sizeof(Tnod));
printf(  "Introdu cheia=   ");
scanf(  "%d  ", &q->cheie);
printf(  "Introdu informatia=   ");
fflush(stdin);
scanf(  "%s  ", q->info);
q->urm=p;
p=q;
}
```

Funcţia de mai sus va realiza:

(a) crearea unei liste liniare simplu înlănţuită desemnată de p cu inserare in faţa listei;

(b) crearea primului nod al listei;

(c) crearea unei liste cu elementele în ordinea inversă a cheilor date;

(d) inserarea unui nod în faţa listei desemnată de p;

(e) crearea unei liste cu inserare în spatele listei.

9. Se consideră descrierile în limbajul C:

```
struct nod {
int cheie;
char info[10];
struct nod *urm;
};
typedef struct nod Tnod;

typedef Tnod * ref;
ref p,r;
/ * p - pointerul spre inceputul listei */
int k;
```

Funcţia următoare:

```
void inserare(void) {
ref s;
s = (ref)malloc(sizeof(Tnod));
*s = *r; r->urm = s;
printf(  "Introdu cheia=   ");
scanf(  "%d  ",&r->cheie);
printf(  "Introdu informatia=   ");
fflush(stdin);
  scanf(  "%s  ", r->info);
}
```

realizează inserarea unui nou nod în faţa unui nod de cheie dată-k, într-o listă liniară simplu înlănţuită, cu structura de mai sus:

(a) chiar dacă lista este vidă;

(b) cu excepţia cazului când lista este vidă;

(c) în faţa nodului de cheie k, de adresă memorată în r, când există în listă un nod cu cheia dată;

(d) după nodul de cheie k, de adresă memorată în r, când există în listă un nod cu cheia dată;

10. Se consideră descrierile în limbajul C:

```
struct nod {
 int cheie;
char info[10];
struct nod *urm;
};
typedef struct nod Tnod;

typedef Tnod * ref;
ref p,q;
/ * p - pointerul spre inceputul listei */
int k;
```

Se dă funcţia:

```
void cautare(void) {
int b = 0;
q=p;
while ((b==0) && ( q != NULL))
if (q->cheie ==k)
b=1;
  else
q=q->urm;
 }
```

care realizează căutarea unui nod de cheie k intr-o listă liniară simplu înlănţuită. Variabila b trebuie introdusă în această funcţie pentru:

(a) dereferenţierea unui nod de adresă NULL;

(b) a evita dereferenţierea unui nod de adresă NULL;

(c) a parcurge mai uşor lista;

(d) a putea stabili dacă nodul există sau nu în listă.

11. Fie p un pointer către primul nod al unei liste liniare simplu înlănţuită de numere întregi cu structura:

```
struct nod {
 int info;
 struct nod *urm;
};

typedef struct nod Tnod ;
```

```
typedef Tnod *ref;
ref p,r;
```

Spre al câtelea nod va pointa r în urma execuţiei secvenţei următoare, presupunând că lista are cel putin 7 elemente:

```
i=1;
r=p;
while(i<=5) {r=r->urm; i++;}
```

(a) 3;
(b) 4;
(c) 5;
(d) 6;
(e) secvenţa este greşită

12. Fie p un pointer către primul nod al unei liste liniare simplu înlănţuită de numere intregi cu structura:

```
struct nod {
 int info;
 struct nod *urm;
};

typedef struct nod Tnod ;

typedef Tnod *ref;
ref p;
/ * p - pointerul spre primul element al listei */
```

iar lista conţine în ordine numerele: 3, 4, 5, 6. Ce valoare va returna apelul functie(p) aceasta avand urmatoarea definitie:

```
int functie (ref p) {
 int s=0;
 ref r;
 r=p;
 while(r->urm) {
 if(!(r->info%2))
 s+=r->info;
 r=r->urm;
 } return s;
}
```

(a) 3;

(b) 4;

(c) 8;

(d) 10;

(e) 18.

13. Fie p un pointer către primul nod al unei liste liniare simplu înlănţuite cu structura:

```
struct nod {
 int info;
 struct nod *urm;
};

typedef struct nod Tnod ;

typedef Tnod *ref;
ref p;
```

Presupunând că lista are cel puţin 3 elemente, care din următoarele instrucţiuni afisează numărul conţinut în al treilea nod din listă?

(a) `printf("%d",p->urm->urm->info);`

(b) `printf("%d",p->urm->info);`

(c) `printf("%d",p->urm->info->info);`

(d) `printf("%d",p->urm->urm->urm);`

(e) `printf("%d",p->urm->urm->urm->info);`

14. Fie p un pointer către primul nod al unei liste liniare simplu înlănţuită de numere întregi cu structura:

```
struct nod {
int info;
struct nod *urm;
};

typedef struct nod Tnod ;

typedef Tnod *ref;
ref p;
```

Care este acţiunea apelului funcţiei următoare? void

```
func(void) {
 ref q;
 q=(ref)malloc(sizeof(Tnod));
 printf("Introduceti informatia:");
 scanf("%d",&(q->info)); q->urm=p;
 p=q;
}
```

(a) crează un nou nod şi îl inserează la sfârşitul listei;

(b) crează un nou nod şi îl inserează la începutul listei;

(c) modifică conţinutul primului nod al listei, adresa rămânând neschimbată;

(d) modifică adresa primului nod, conţinutul rămâne neschimbatş

(e) nici una din acţiunile de mai sus.

15. Fie p un pointer către primul nod al unei liste liniare simplu inlănţuită de numere întregi cu structura:

```
struct nod {
int info;
struct nod *urm;
};

typedef struct nod Tnod ;

typedef Tnod *ref;
ref p,r,s;
/ * p indica primul nod al listei, r indica un nod oarecare al listei*/
```

Care este acţiunea apelului funcţiei operatie?

```
 void operatie(void)
{
  if (r->urm==NULL)
if (p==r) {
  p=NULL;
  free(r);
  } else {
  s=p;
  while (s->urm!=r) s=s->urm;
  s->urm=0;
  free(r);
  }
```

```
  else {
  s=r->urm;
  *r=*s;
  free(s);
 } }
```

(a) suprimarea nodului de după nodul precizat de r;

(b) suprimarea nodului precizat de r;

(c) suprimarea unui nod diferit de ultimul nod al listei.

16. Fie p un pointer către primul nod al unei liste liniare simplu înlănţuită de numere întregi, q un pointer către ultimul nod al aceleeaşi liste, listă care are structura:

```
 struct nod {
  int info;
  struct nod *urm;
 };

 typedef struct nod Tnod ;

 typedef Tnod *ref;
 ref p,q;
```

Care este efectul apelului funcţiei următoare?

```
 void func(void)
{
  ref r;
  r=(ref)malloc(sizeof(Tnod));
printf("Introduceti informatia:");
  scanf("%d",&(r->info));
r->urm=NULL;
q->urm=r;
q=r;
 }
```

(a) crează un nou nod şi îl inserează la sfârşitul listei desemnată de p, nevidă, care are ultimul nod de adresă q;

(b) crează un nou nod şi îl inserează la începutul listei;

(c) modifică conţinutul ultimului nod al listei, adresa rămânând neschimbată;

(d) modifică adresa ultimului nod, conţinutul rămâne neschimbat;

(e) nici una din acţiunile de mai sus.

17. Fie q un pointer către ultimul nod al unei liste liniare simplu înlănţuită de numere întregi cu structura:

```
 struct nod {
  int info;
struct nod *urm;
 }

 typedef struct nod Tnod ;

 typedef Tnod *ref;
 ref q;
```

Care este acţiunea funcţiei următoare?

```
 void func(void)
{
 ref r;
 r=(ref)malloc(sizeof(Tnod));
 printf("Introduceti informatia:");
 scanf("%d",&(r->info)); r->urm=NULL;
 if(q!=NULL)
    q->urm=r;
 q=r;
 }
```

(a) crează un nou nod şi îl inserează la sfârşitul listei;

(b) crează un nou nod şi îl inserează la începutul listei;

(c) modifică conţinutul ultimului nod al listei, adresa rămânând neschimbată;

(d) modifică adresa ultimului nod, conţinutul rămâne neschimbată;

(e) nici una din acţiunile de mai sus.

18. Fie p un pointer către primul nod al unei liste liniare simplu înlănţuită de numere întregi cu structura:

```
 struct nod {
  int info;
  struct nod *urm;
 };

 typedef struct nod Tnod ;
```

```
typedef Tnod *ref;
ref p;
```

Care este acţiunea funcţiei următoare?

```
ref func(int k) {
ref r;
int b=0;
r=p;
while((b==0)&&(r!=NULL)) {
if(r->info==k)
b=1;
else r=r->urm;
}
return r;
}
```

(a) caută primul nod al listei şi returnează valoarea sa;

(b) caută nodul care conţine valoarea k în câmpul info şi returnează adresa sa;

(c) caută nodul următor nodului ce conţine valoarea k în câmpul info;

(d) caută nodul de pe poziţia k şi returnează adresa sa;

(e) nici una din acţiunile de mai sus.

19. Se consideră descrierile în limbajul C:

```
struct nod {
int cheie;
char info[10];
struct nod *urm;
};
typedef struct nod Tnod;

typedef Tnod * ref;
ref p; / * p - pointerul spre \^{\i}nceputul listei */
```

Descrierea anterioară se poate utiliza pentru a defini o structură de date de tip

(a) listă liniară dublu înlănţuită cu elemente care au cheia de tip întreg;

(b) listă circulară simplu înlănţuită cu elemente care au cheia de tip întreg;

(c) listă circulară dublu înlănţuită cu elemente care au cheia de tip întreg;

(d) arbore binar cu elemente care au cheia de tip întreg;

(e) listă liniară simplu înlănţuită cu elemente care au cheia de tip întreg.

20. Se consideră descrierile în limbajul C:

```
struct nod {
int info;
struct nod *urm;
};

typedef struct nod Tnod ;

typedef Tnod *ref;
ref p,r,s;
/ * p indica primul nod al listei, r indica predecesorul nodului
    care va fi suprimat, s o variabila auxiliara*/

void suprimare(void) {
if (r->urm==NULL)
printf(  "Eroare: nodul de suprimat nu exista.\n  ");
else {
s=r->urm;
...
} }
```

Care din secvenţele următoare pot completa descrierea corectă a funcţiei care va realiza suprimarea succesorului nodului precizat de r?

(a) `r->urm=s; free(s);`

(b) `r->urm=s->urm; free(s);`

(c) `r->urm=s->urm; free(r);`

21. O lista liniară dublu înlănţuită, cu elemente întregi, este descrisă in C, astfel:

(a)
```
struct nod {
int element;
struct nod *urm, *ant;
};
typedef struct nod Tnod;
typedef Tnod *ref;
ref p;
```

(b)
```
struct nod {
int element;
struct nod urm, ant;
};
typedef struct nod *Tnod;
typedef Tnod ref;
ref p;
```

(c)
```
typedef struct nod {
int element;
struct nod *urm, *pred;
}Tnod;
typedef Tnod *ref;
ref p;
```

(d)
```
struct nod {
int element;
struct nod *urm, *pred;
};
typedef struct nod Tnod;
typedef Tnod ref;
ref p;
```

22. Se consideră descrierea în C, a unei structuri de tip listă liniară dublu înlănţuită :

```
struct nod {
int info;
struct nod *ant,*urm;
};
typedef struct nod Tnod;
typedef Tnod *ref;
ref p;
```

Care din afirmaţiile următoare sunt greşite:

(a) info este câmpul de informaţie utilă (un număr întreg);

(b) existenţa celor doi pointeri permite parcurgerea listei în ambele sensuri;

(c) o listă dublu înlănţuită permite inserarea unui nou element doar la unul din capetele listei;

(d) toate afirmaţiile de mai sus sunt adevărate.

23. Se consideră descrierile în limbajul C:

```
struct nod {
int element;
```

```
 struct nod *urm, *ant;
 };
 typedef struct nod Tnod;
 typedef Tnod *ref;
 ref p,r;
```

Funcţia următoare:

```
 void inserare(void) {
  ref pred, s;
  pred = r->ant;
s=(ref)malloc(sizeof(Tnod));
  printf(  "Introdu elementul=   ");
scanf(  "%d  ", &s->element);
  s->ant = pred;
s->urm = r;
  pred->urm = s;
  r->ant = s;
 }
```

realizează:

(a) inserarea unui nou nod înaintea nodului de adresă `r`;

(b) inserarea unui nou nod după nodul de adresă `r`;

(c) inserarea unui nou nod înaintea nodului de adresă `r`, cu `r->ant!=0`;

(d) inserarea unui nou nod înaintea nodului `*r`;

24. Se consideră descrierile în limbajul C:

```
 struct nod {
 int element;
 struct nod *urm, *ant;
 };
 typedef struct nod Tnod;
 typedef Tnod *ref;
 ref p,r;
 / * p desemneaza primul nod al listei, iar r nodul care trebuie sters*/
 void sterg_nod(void) {
 ref pred,suc;
 pred=r->ant;
 suc=r->urm;
 ...
```

```
if (r==p) / * daca nodul de suprimat este primul nod */
p=p->urm;
free(r);
}
```

Cu care din secvenţele următoare trebuie completată funcţia sterg_nod astfel încât să realizeze suprimarea unui nod de adresă r, dintr-o listă liniară dublu înlănţuită cu structura nodurilor definită mai sus:

(a)
```
if (r->urm!=NULL) suc->ant=pred;
if (r->ant!=NULL) pred->urm=suc;
```

(b)
```
if (r->ant!=NULL) pred->urm=suc;
if (r->urm!=NULL) suc->urm=pred;
```

(c)
```
if (r->urm!=NULL) pred->urm=suc;
if (r->ant!=NULL) suc->ant=pred;
```

25. O stivă este:

    (a) listă liniară în care inserările se fac la un capăt al listei şi suprimările la celalalt capăt al listei;

    (b) o listă liniară de tipul LIFO (Last In First Out);

    (c) o listă liniară cu restricţie la intrare;

    (d) o listă liniară în care se manipulează mereu elementul cel mai recent introdus.

26. Se considera o stivă implementată prin tipul tablou:

```
#define lung_max 100
typedef int nod;
typedef struct {
 int varf;
nod elemente [lung_max];
} stiva;
stiva s;
```

Care din urmatoarele funcţii realizează iniţializarea stivei s (iniţializarea constă în a face stiva vidă), stiva crescând, în ordinea descreşterii indexului în tablou.

(a)
```
void initializare (stiva s) {
s->varf = lung_max; }
```

(b)
```
void initializare(stiva *s) {
s->varf = NULL;
}
```

(c)
```
void initializare(stiva *s) {
s->varf = lung_max;
}
```

(d)
```
void initializare () {
s.varf = lung_max;
}
```

27. Se consideră descrierile în limbajul C:

```
typedef int tipelem;
typedef struct nod {
 tipelem elem;
struct nod *urm;
} Tnod;

typedef Tnod *ref;
ref varf; / * virful stivei */
ref r; / * variabila auxiliara */
void adauga(void) {
 r=malloc(sizeof(Tnod));
 printf(  "Introduceti informatia(nr.intreg,max 5 cifre):  ");
 scanf(  "%d  ",&r->elem);
 ...
}
```

Care din secvenţele următoare completează corect funcţia adauga astfel încât ea să realizeze adăugarea unui element în stiva s?

(a)
```
r.urm=varf;
varf=r;
```

(b)
```
r->urm=varf;
varf=r;
```

(c)
```
r->urm=varf;
```

28. O structură de date de tip coadă este:

(a) structură de tip listă cu restricţie la intrare;

(b) o listă liniară de tipul FIFO (First In First Out);

(c) o lista liniară în care toate operaţiile se efectuează doar la unul din capetele listei;

(d) o listă liniară în care inserările se efectuează la un capăt al listei, iar suprimările şi ori ce alt acces se efectuează la celălalt capăt al listei.

29. Se consideră descrierile în limbajul C:

```
 typedef int tipelement;

 typedef struct nod {
tipelement element;
struct nod *urm;
 }Tnod;

 typedef Tnod *ref;

 typedef struct {
  ref fata,spate;
 }Coada;
 Coada c;
 void initializare(Coada* c) { / * creeaza nodul fictiv*/
  c->fata=(ref)malloc(sizeof(Tnod));
c->fata->urm=0;
        / * nodul fictiv este primul si ultimul nod*/
c->spate=c->fata;
 } / * initializare */
```

Pentru iniţializarea cozii c, cu structura de mai sus, cu element fictiv cap de listă, se va apela funcţia iniţializare astfel:

(a) initializare(c);

(b) initializare(&c);

(c) initializare(* c).

30. Se consideră descrierile în limbajul C:

```
 typedef int tipelement;
 typedef struct nod {
 tipelement element;
 struct nod *urm;
 } Tnod;

 typedef Tnod *ref;

 typedef struct {
 ref fata,spate;
 } Coada;
```

```
 Coada c;
 void adauga(tipelement x, Coada * c) {
c->spate->urm=malloc(sizeof(Tnod));
  ...
  c->spate->element=x;
c->spate->urm=NULL;
 } / * adauga */
```

Cu care din secvenţele următoare trebuie completată funcţia adauga pentru a realiza adăugarea unui element de cheie dată x într-o coadă implementată prin structura de mai sus, cu element cap de listă.

(a) `c->fata= c->spate->urm;`

(b) `c->spate=c->spate->urm;`

(c) `c->spate->urm=NULL;`

4 Teoria grafurilor şi combinatorică

1. Ce rang are permutarea $\langle 6, 1, 3, 2, 5, 4\rangle$ în enumerarea lexicografică a permutărilor?

   (a) 411
   (b) 412
   (c) 605
   (d) 607

2. Câte permutări ale literelor ABCDEFGH conţin subşirul BCD?

   (a) 6!
   (b) 8!
   (c) 8!/3
   (d) 2^5

3. Câte şiruri se pot obţine rearanjând literele şirului SUCCESS?

   (a) 7!
   (b) 2^7
   (c) 420
   (d) 12

4. Câte numere întregi cuprinse între 1 şi 1000 se divid cu 7, dar nu se divid cu 3?

   (a) 93
   (b) 95
   (c) 92
   (d) 136

5. Într-un sertar sunt opt ciorapi maro şi 12 ciorapi negri. Un copil scoate la întâmplare pe întuneric ciorapi din sertar. Câţi ciorapi trebuie să scoată copilul pentru a fi sigur că a scos cel puţin 2 ciorapi negri?

   (a) 3
   (b) 10
   (c) 14
   (d) 95

6. Ce coeficient are $x^8\, y^2$ în $(x + 2 \cdot y)^{10}$?

   (a) 1

(b) 45

(c) 180

(d) 5120

(e) 960

7. Câte submulţimi cu mai mult de 2 elemente are o mulţime cu 6 elemente?

   (a) 32

   (b) 61

   (c) 42

   (d) 21

8. Care este forma generală a relaţiei de recurenţă

$$a_n = -3\,a_{n-1} - 3\,a_{n-2} - a_{n-3}$$

   cu condiţiile iniţiale $a_0 = 1, a_1 = -2$ şi $a_2 = -1$?

   (a) $a_n = 2\,n^2 - 5\,n + 1$

   (b) $a_n = -\frac{1}{2}n^3 + \frac{7}{2}n^2 - 6\,n + 1$

   (c) $a_n = -4 \cdot (-2)^n + (5n + 5)(-1)^n$

   (d) $a_n = (1 + 3\,n - 2\,n^2) \cdot (-1)^n$

9. Un mesaj transmis pe un canal de comunicare este o secvenţă de 2 tipuri de semnale: semnale de tip A care durează 1 microsecundă, şi semnale de tip B care durează 2 microsecunde. De exemplu, mesajul $ABAAB$ durează $3 \times 1 + 2 \times 2 = 7$ microsecunde.

   Fie a_n numărul de mesaje diferite care durează n microsecunde. Care este valoarea lui a_{10}?

   (a) 89

   (b) 55

   (c) 2917

   (d) 144

10. In internet, format din reţele interconectate de calculatoare, fiecărei conexiuni de reţea dintr-un calculator i se atribuie o adresă internet. Protocolul IPv4 prevede că o adresă internet este un şir de 32 biţi, format din un număr de reţea (*netid*) urmat de un număr de gazdă (*hostid*). Sunt 3 tipuri de adrese internet:

    (a) Clasa A: acestea sunt de forma $0\,\underbrace{b_1 b_2 b_3 b_4 \ldots b_7}_{\text{netid}}\,\underbrace{b_8 b_9 \ldots b_{30} b_{31}}_{\text{hostid}}$

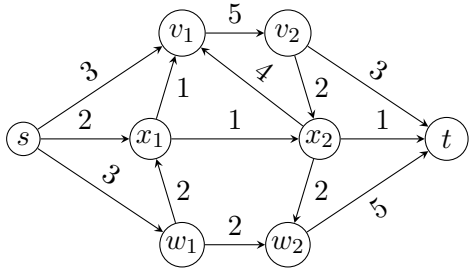
    (b) Clasa B: acestea sunt de forma $10\,\underbrace{b_2 b_3 b_4 \ldots b_{15}}_{\text{netid}}\,\underbrace{b_{16} b_{17} \ldots b_{30} b_{31}}_{\text{hostid}}$

(c) Clasa C: acestea sunt de forma $110\underbrace{b_3b_4b_5\ldots b_{23}}_{\text{netid}}\underbrace{b_{24}b_{25}\ldots b_{30}b_{31}}_{\text{hostid}}$

Câte adrese IPv4 diferite sunt disponibile pentru conexiunile de reţea din internet?

(a) $2^{29} \cdot 2^{30} \cdot 2^{31}$
(b) 2^{31}
(c) $7 \cdot 2^{29}$
(d) 2^{32}

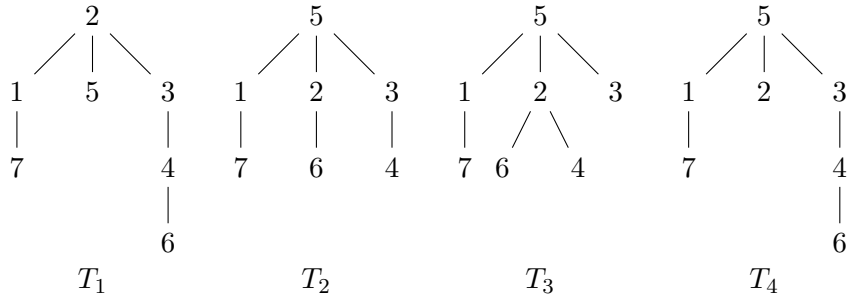
11. Fie reţeaua de transport G cu sursa s şi destinaţia t:



Care este valoarea fluxului maxim în G?

(a) 8
(b) 5
(c) 6
(d) 7

12. Care din arborii următori are secvenţa Prüfer 5,4,3,5,1?

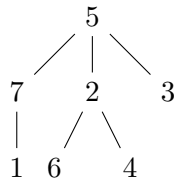


(a) T_1
(b) T_2

(c) T_3

(d) T_4

13. Care este secvenţa Prüfer a arborelui următor:



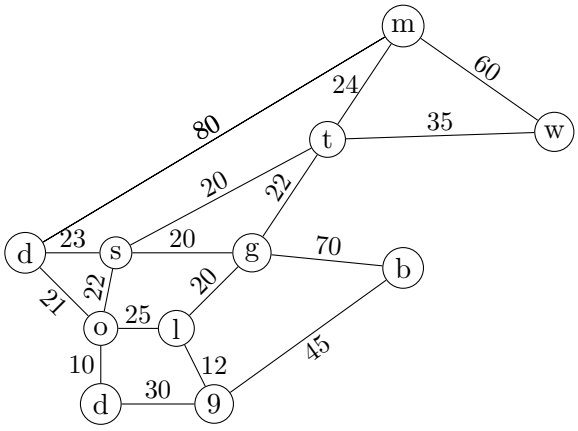
(a) $7, 5, 2, 2, 5$

(b) $1, 3, 4, 5, 2$

(c) $7, 2, 5, 5, 5$

(d) $3, 4, 6, 2, 2$

14. Fie graful ponderat



Ce greutate totală are arborele minim de acoperire al acestui graf?

(a) 229

(b) 216

(c) 230

(d) 234

15. Care este semnificaţia lui $\left\{ {n \atop k} \right\}$?

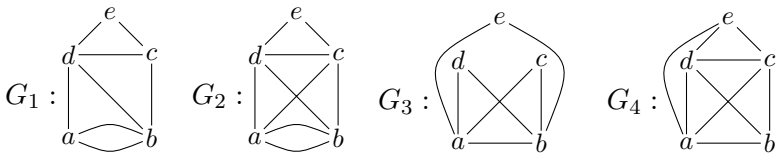
a) Numărul de submulţimi cu k elemente al unei mulţimi cu n elemente.

b) Numărul de k-permutări al unei mulţimi cu n elemente.

c) Numărul de feluri în care se poate partiţiona o mulţime cu n elemente în exact k submulţimi nevide şi disjuncte.

d) Numărul de feluri în care pot fi puse n persoane la k mese rotunde identice, astfel încât nici o masă rotundă să nu rămână neocupată.

16. Care din grafurile următoare este eulerian?



(a) G_1, G_4

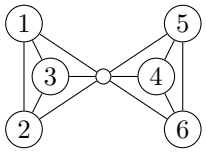
(b) G_2, G_3

(c) nici unul

(d) G_4

(e) G_1, G_3

17. Fie configuraţia următoare:



Câte colorări diferite cu 3 culori are această configuraţie?

(a) 24

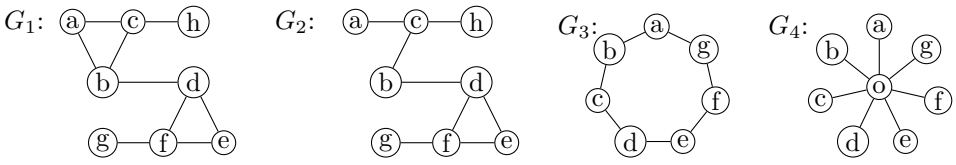
(b) 1120

(c) 120

(d) 64

(e) 216

18. Care din grafurile următoare are un cuplaj perfect?



(a) G_1

(b) G_2

(c) G_3

(d) G_4

(e) 216

19. Care din familiile următoare de mulţimi are un sistem de reprezentanţi distincţi:

(a) $\{1\}, \{1,2\}, \{2\}, \{3,4,5\}, \{1,2,3,4,5\}$

(b) $\{1,2\}, \{2,3\}, \{1,2,3\}, \{1,3\}, \{1,2,3,4\}, \{1,2,4,5\}$

(c) $\{1,2,3\}, \{2,3,4\}, \{3,4,5\}, \{4,5\}, \{1,2,5\}$

(d) $\{1,2,5\}, \{1,5\}, \{1,2\}, \{2,5\}, \{1,3\}, \{6\}, \{4,6,7\}$

20. Câţi arbori diferiţi cu 5 noduri, numerotate de la 1 la 5, există?

(a) 10

(b) 273

(c) 32

(d) 120

(e) 125

21. Fie G un graf cu n noduri. Să se indice toate afirmaţiile care sunt echivalente cu faptul că G este un arbore:

1. G este un graf conectat fără cicluri.
2. G are $n-1$ muchii.
3. G nu are cicluri, iar dacă se adaugă o muchie la G, se crează exact un ciclu.
4. Fiecare pereche de noduri din G este conectată cu exact o cale simplă.

(a) $1, 2, 3, 4$

(b) $1, 2, 3$

(c) $1, 3, 4$

(d) 1

5 Limbaje formale şi teoria automatelor

1. Limbajul generat de gramatica $G = (V_N, V_T, S, P)$, unde $V_N = \{S\}$; $V_T = \{0, 1, 2\}$; $P = \{S \to 0S0|1S1|2S2|0\}$, este:

   (a) $L = \{0^n 1^n 2^n | n \geq 1\}$;

   (b) $L = \{w \in \{0, 1, 2\}^* | w$ palindrom centrat în $0\}$;

   (c) $L = \{0^n 1^m 2^p | n, m, p \in N\}$;

   (d) $L = \{w0x | w, x \in \{0, 1, 2\}^*, x$ oglinditul lui $w\}$;

2. Limbajul generat de gramatica $G = (V_N, V_T, S, P)$, unde $V_N = \{S\}$; $V_T = \{a, b\}$; $P = \{S \to aSb|ab\}$, este:

   (a) $L = \{a^n b^m | n, m \geq 1\}$;

   (b) $L = \{w \in \{a, b\}^* | w$ conţine infixul $ab\}$;

   (c) $L = \{a^n b^n | n \geq 0\}$;

   (d) $L = \{a^n b^n | n \geq 1\}$;

3. Limbajul generat de gramatica $G = (V_N, V_T, S, P)$, unde $V_N = \{S\}$; $V_T = \{a, b\}$; $P = \{S \to bSb|bAb, A \to aA|aa\}$, este:

   (a) $L = \{b^n aab^n | n \geq 1\}$;

   (b) $L = \{b^n aaa^m b^n | n \geq 1, m \geq 0\}$;

   (c) $L = \{b^n (aa)^m b^n | n, m \geq 1\}$;

   (d) $L = \{b^n a^m b^n | n \geq 1, m \geq 2\}$;

4. Limbajul generat de gramatica $G = (V_N, V_T, S, P)$, unde $V_N = \{S\}$; $V_T = \{a, b\}$; $P = \{S \to aSb|aAb, A \to aA|\lambda\}$, este:

   (a) $L = \{a^m b^n | m \geq n \geq 1, \}$;

   (b) $L = \{a^n a^m b^n | n \geq 0, m \geq 0\}$;

   (c) $L = \{a^{i+1} b^i | i \geq 1\}$;

   (d) $L = \{a^{n+i} b^n | n \geq 1, i \geq 0\}$;

5. Limbajul generat de gramatica $G = (V_N, V_T, S, P)$, unde $V_N = \{S, A, B\}$, $V_T = \{a, b, c\}$, $P = \{S \to abc|aAbc, Ab \to bA, Ac \to Bbcc, bB \to Bb, aB \to aaA|aa\}$, este:

   (a) $L = \{a^n b^m c^p | n, m, p \geq 1\}$;

   (b) $L = \{w \in \{a, b, c\}^* | w$ conţine cel puţin trei litere $\}$;

   (c) $L = \{w \in \{a, b, c\}^* | w$ palindrom centrat in $abc\}$;

   (d) $L = \{a^n b^n c^n | n \geq 1\}$;

6. Limbajul generat de gramatica $G = (V_N, V_T, S, P)$, unde $V_N = \{S, A\}$, $V_T = \{a, b, c\}$, $P = \{S \to A|aS|bS|cS, A \to Aa|Ab|Ac|\lambda\}$, este :

   (a) $L = \{a^n b^m c^p | n, m, p \geq 0\}$;

   (b) $L = \{w \in \{a, b, c\}^* | w$ conţine cel puţin o literă $\}$;

   (c) $L = \{a, b, c\}^*$;

   (d) $L = \{a^n b^n c^n | n \geq 1\}$;

7. Limbajul generat de gramatica $G = (V_N, V_T, S, P)$, unde $V_N = \{S, A, B, C, D, E, F\}$, $V_T = \{a, b, \ldots, z\}$, $P = \{S \to iA|aB, A \to oC, B \to nD, C \to n, D \to a\}$, este :

   (a) $L = \{ion, ana\}$;

   (b) $L = \{ion, ani, ina, aia, iao\}$;

   (c) $L = \{w \in \{a, b, \ldots, z\}^* | w$ începe cu a sau i şi se termină cu a sau n $\}$;

8. Limbajul generat de gramatica $G = (V_N, V_T, S, P)$, unde $V_N = \{S\}$, $V_T = \{a, b\}$, $P = \{S \to aS|bS|a\}$, este :

   (a) $L = \{a^n a, b^n a | n \geq 0\}$;

   (b) $L = \{wa | w \in \{a, b\}^*\}$;

   (c) $L = \{w \in \{a, b\}^* | w$ se termină cu a $\}$;

   (d) $L = \{w \in \{a, b\}^* | w$ începe cu a $\}$;

9. Limbajul generat de gramatica $G = (V_N, V_T, S, P)$, unde $V_N = \{S, X\}$, $V_T = \{a, b, c\}$, $P = \{S \to aS|bS|cS|abbaX, X \to aX|bX|cX|\lambda\}$, este:

   (a) $L = \{w \in \{a, b, c\}^* | w$ conţine abba $\}$;

   (b) $L = \{wabbaw | w \in \{a, b, c\}^*\}$;

   (c) $L = \{w \in \{a, b, c\}^* | w$ se termină cu abba $\}$;

   (d) $L = \{wabbax | w, x \in \{a, b, c\}^*\}$;

   (e) $L = \{w \in \{a, b, c\}^* | w$ începe cu abba $\}$;

10. Regulile gramaticii $G = (V_N, V_T, S, P)$, unde $V_N = \{S\}$, $V_T = \{PCR, PDAR, UDMR\}$, $P = \{S \to PCR|PDAR|UDMR\}$, respectă restricţiile impuse gramaticilor:

    (a) regulate (tip 3);

    (b) independente de context (tip 2);

    (c) dependente de context (tip 1);

    (d) de tipul 0, 1, 2, 3;

11. Regulile gramaticii $G = (V_N, V_T, S, P)$, unde $V_N = \{S, X\}$, $V_T = \{a, b, c\}$, $P = \{S \to aS|bS|cS|abbaX, X \to aX|bX|cX|\lambda\}$, respectă restricţiile impuse gramaticilor:

(a) regulate (tip 3);

(b) independente de context (tip 2);

(c) dependente de context (tip 1);

(d) de tipul 0, 1, 2, 3;

12. Regulile gramaticii $G = (V_N, V_T, S, P)$, unde $P = \{S \to abc|aAbc,\ Ab \to bA,\ Ac \to Bbcc,\ bB \to Bb,\ aB \to aaA|aa\}$, respectă restricţiile impuse gramaticilor:

(a) regulate (tip 3);

(b) independente de context (tip 2);

(c) dependente de context (tip 1);

(d) de tipul 0;

13. Regulile gramaticii $G = (V_N, V_T, S, P)$, unde $P = \{S \to aSb|\lambda\}$, respectă restricţiile impuse gramaticilor:

(a) regulate (tip 3);

(b) independente de context (tip 2);

(c) dependente de context (tip 1);

(d) de tipul 0;

14. Regulile gramaticii $G = (V_N, V_T, S, P)$, unde $P = \{S \to aS|bS|\lambda\}$, respectă restricţiile impuse gramaticilor:

(a) regulate (tip 3);

(b) independente de context (tip 2);

(c) dependente de context (tip 1);

(d) de tipul 0;

15. Expresia regulată ce notează limbajul $L = \{w|$ şiruri de 0 şi 1 terminate cu 1 $\}$, este:

(a) $(0|1)^*1$;

(b) $1|(0|1)^*$;;

(c) $(0^*|1^*)^*1$;

(d) $(1|0|1)^*1$;

16. Expresia regulată ce notează limbajul $L = \{w|$ şiruri de 0 şi 1 ce conţin cel putin un simbol 1 $\}$, este:

(a) $(0|1)^*1$;

(b) $1|(0|1)^*1(1|0)^*$;

(c) 0^*11^*;

(d) $(0|1)^*1(0|1)^*$;

17. Expresia regulată ce notează limbajul $L = \{w|$ şiruri de 0 şi 1 ce conţin cel putin un simbol $\}$, este:

(a) $(0|1)^*1$;

(b) $1|0|(0|1)^*$;

(c) $(0^*|1^*)^*|1|0$;

(d) $(1|0)(0|1)^*$;

18. Expresia regulată ce notează limbajul $L = \{ana, ani, ina, ini\}$, este:

(a) $ana|ani|ina|ini$;

(b) $(a|i)n(a|i)$;

(c) a^*ni^*;

(d) $(a|i)^*n(a|i)^*$;

19. Expresiile regulate notează:

(a) limbajele regulate;

(b) limbajele de tipul 0 şi 3;

(c) limbajele recunoascute de automate finite deterministe;

(d) numai limbajele finite;

20. Limbajul L notat de expresia regulata $(a|i)n(a|i)$ este:

(a) $L = \{ana, ani, ina, ini\}$;

(b) $L = \{w \in \{a, n, i\}^*|w$ începe şi se termină cu a sau i$\}$;

(c) $L = \{w \in \{a, n, i\}^*|w$ începe cu a sau i şi se termină cu na sau ni$\}$;

(d) $L = \{w \in \{a, n, i\}^*|w$ începe şi se termină cu a sau i, iar litera n este in mijlocul cuvântului w $\}$;

21. Limbajul L notat de expresia regulata $(0|1)(0|1)^*$ este:

(a) $L = \{0, 1, 00, 01, 10, 11, 000, \ldots\}$;

(b) $L = \{w \in \{0, 1\}^*|w$ începe cu 0 sau 1$\}$;

(c) $L = \{0, 1\}^*$;

22. Limbajul L notat de expresia regulata $01^*|1$; este:

(a) $L = \{0, 1, 00, 01, 10, 11, 000, \ldots\}$;

(b) $L = \{w \in \{0,1\}^* | w$ începe cu 1 $\}$;

(c) $L = \{w \in \{0,1\}^* | w$ începe cu 1 $\}$;

(d) $L = \{01^n | n \geq 1\} \bigcup \{1\}$;

23. Limbajul L notat de expresia regulata $(11)^*1$ este:

(a) $L = \{1, 11, 111, 1111, \ldots\}$;

(b) $L = \{w \in \{0,1\}^* | w$ are un număr impar de simboluri $1\}$;

(c) $L = \{1w | w = 1^{2n}, n \in N\}$;

(d) $L = \{1^{2n+1}, n \in N\}$;

24. Limbajul L notat de expresia regulata $(1|0)^*0(0|1)$ este:

(a) $L = \{0, 1, 00, 01, 10, 11, \ldots\}$;

(b) $L = \{w \in \{0,1\}^* | w$ se termină cu 00 sau $01\}$;

(c) $L = \{w \in \{0,1\}^* | w$ are cel putin un simbol 0 $\}$;

(d) $L = \{w \in \{0,1\}^* | w$ are 0 în penultima poziţie $\}$;

6 Răspunsuri

Algoritmica

1. 1c
2. 2d
3. 3d
4. 4c,4d
5. 5a,5c
6. 6c
7. 7c
8. 8b
9. 9a,9d
10. 10d
11. 11b,11e
12. 12d
13. 13b
14. 14c,14d,14e
15. 15b
16. 16b,16c
17. 17a,17b
18. 18c
19. 19d
20. 20c
21. 21b
22. 22b,22c
23. 23b
24. 24b,24d
25. 25a,25b,25d
26. 26b,26e
27. 27d,27f
28. 28d
29. 29b,29d,29f
30. 30b

Logică Computaţională

1. 1c,1d
2. 2a,2c
3. 3a,3c,3d
4. 4a,4b
5. 5b,5c
6. 6a,6c
7. 7a,7b,7c
8. 8b
9. 9a,9b,9c,9d
10. 10c,10d
11. 11b
12. 12c,12d
13. 13a,13c,13d
14. 14b,14c
15. 15b
16. 16a
17. 17b
18. 18b,18c
19. 19c
20. 20a

Structuri de date

1. 1b,1c,1e
2. 2b
3. 3a,3c,3d,3e
4. 4a
5. 5b
6. 6c
7. 7a,7c
8. 8a,8b,8d
9. 9c
10. 10b
11. 11d
12. 12b
13. 13a
14. 14b
15. 15b
16. 16a
17. 17a
18. 18b
19. 19b,19e
20. 20b
21. 21a,21c
22. 22c,22d
23. 23c
24. 24a
25. 25b,25d
26. 26d
27. 27b
28. 28b,28d
29. 29b
30. 30b

Teoria grafurilor şi combinatorică

1. 1d
2. 2a
3. 3c
4. 4b
5. 5b
6. 6c
7. 7c
8. 8d
9. 9a
10. 10c
11. 11d
12. 12d
13. 13a
14. 14a
15. 15c
16. 16b
17. 17e
18. 18a
19. 19c
20. 20e
21. 21c

Limbaje formale şi teoria automatelor

1. 1b,1d
2. 2d
3. 3b,3d
4. 4a,4d
5. 5d
6. 6c
7. 7a
8. 8b,8c
9. 9a,9d
10. 10a,10b,10c,10d
11. 11b
12. 12d
13. 13b,13d
14. 14a,14b,14d
15. 15a,15c,15d
16. 16b,16d
17. 17d
18. 18a,18b
19. 19a,19c
20. 20a
21. 21a,21b
22. 22d
23. 23c,23d
24. 24b,24d