

Grile - Modulul 1

1 Algoritmi si structuri de date

1. Se dă următorul algoritm (vezi Algoritmul 1).

Algorithm 1 Algoritmul Ordonare prin metoda bulelor

```
1: procedure SORTARE_BUBBLE( $a, n$ )
2:   for  $i \leftarrow 1, n - 1$  do
3:     for  $j \leftarrow n, i + 1$  STEP  $- 1$  do
4:       if ( $a_j < a_{j-1}$ ) then
5:          $a_{j-1} \leftrightarrow a_j$ 
6:       end if
7:     end for
8:   end for
9: end procedure
```

Care vor fi valorile vectorului a după terminarea pasului $i = 5$, știind că la intrare avem valorile $n = 7$ și $a = (7, 5, 3, 1, 2, 4, 6)$?

- A. $(1, 2, 3, 7, 6, 5, 4)$
- B. $(1, 2, 3, 4, 5, 6, 7)$
- C. $(1, 2, 3, 4, 7, 6, 5)$
- D. $(1, 2, 3, 4, 5, 7, 6)$**
- E. $(1, 5, 3, 2, 4, 6, 7)$

2. Se dă următorul algoritm (vezi Algoritmul 2).

Algorithm 2 Algoritmul subșir crescător maximal

```
1: procedure SUBSIRMAXIM( $n, A$ )
Require:    $\begin{cases} n & \text{lungimea sirului } A \\ a_0, a_1, \dots, a_{n-1} & \text{elementele sirului} \end{cases}$ 
2:    $max\_sir \leftarrow 0, position \leftarrow 0$ 
3:    $lung_{n-1} \leftarrow 1$ 
4:    $where_{n-1} \leftarrow n - 1$ 
5:   for  $i \leftarrow n - 2, 0$  STEP  $- 1$  do
6:      $lung_i \leftarrow 1$ 
7:      $where_i \leftarrow i$ 
8:     for  $j \leftarrow i + 1, n - 1$  do
9:       if ( $(a_i \leq a_j) \wedge (lung_i < lung_j + 1)$ ) then
10:         $lung_i \leftarrow lung_j + 1$ 
11:         $where_i \leftarrow j$ 
12:      end if
13:    end for
14:    if ( $max\_sir < lung_i$ ) then
15:       $max\_sir \leftarrow lung_i$ 
16:       $position \leftarrow i$ 
17:    end if
18:  end for
19:  Output  $\{max\_sir\}$ 
20:   $i \leftarrow position$ 
21:  while ( $i \neq where_i$ ) do
22:    Output  $\{a_i\}$ 
23:     $i \leftarrow where_i$ 
24:  end while
25:  Output  $\{a_i\}$ 
26: end procedure
```

Care vor fi valorile vectorului $lung$ la sfârșitul algoritmului, știind că datele de intrare sunt: $n = 8$ și $a = (4, 7, 9, 5, 6, 2, 4, 8)$?

- A. (4, 2, 1, 3, 2, 2, 3, 1)
- B. (4, 2, 1, 3, 3, 3, 2, 1)
- C. (4, 2, 1, 2, 2, 3, 2, 1)
- D. (4, 2, 1, 3, 2, 3, 2, 1)**
- E. (4, 2, 1, 3, 1, 3, 2, 1)

3. Se dă următorul algoritm (vezi Algoritmul 3).

Algorithm 3 Algoritmul Sortare minim

```

1: procedure SORTARE_MIN( $a, n$ )
2:   for  $i \leftarrow 1, n - 1$  do
3:     for  $j \leftarrow i + 1, n$  do
4:       if ( $a_i > a_j$ ) then
5:          $a_i \leftrightarrow a_j$ 
6:       end if
7:     end for
8:   end for
9: end procedure

```

Care vor fi valorile vectorului a după terminarea pasului $i = 3$, știind că la intrare avem valorile $n = 7$ și $a = (7, 4, 5, 6, 1, 3, 2)$?

- A. (1, 7, 5, 6, 4, 3, 2)
- B. (1, 2, 7, 6, 5, 4, 3)
- C. (1, 2, 3, 4, 7, 6, 5)
- D. (1, 2, 3, 7, 6, 5, 4)**
- E. (1, 2, 5, 4, 6, 7, 3)

4. Se dă următorul algoritm (vezi Algoritmul 4).

Algorithm 4 Algoritmul Ordonare prin numarare

```

1: procedure SORTARE_NUMARARE( $a, n; c$ )
2:   for  $i \leftarrow 1, n$  do
3:      $b_i \leftarrow 0$ 
4:   end for
5:   for  $i \leftarrow 1, n - 1$  do
6:     for  $j \leftarrow i + 1, n$  do
7:       if ( $a_i < a_j$ ) then
8:          $b_j \leftarrow b_j + 1$ 
9:       else
10:         $b_i \leftarrow b_i + 1$ 
11:      end if
12:    end for
13:  end for
14:  for  $i \leftarrow 1, n$  do
15:     $c_{b_i+1} \leftarrow a_i$ 
16:  end for
17: end procedure

```

Care vor fi valorile vectorului b după terminarea pasului $i = 3$ (din instrucțiunea repetitivă 5-13), știind că la intrare avem valorile $n = 7$ și $a = (2, 4, 5, 6, 7, 1, 3)$?

- A. (1, 3, 2, 2, 2, 0, 1)
- B. (1, 3, 4, 5, 4, 0, 1)

C. (1, 3, 4, 3, 2, 0, 1)

D. (1, 3, 4, 3, 3, 0, 1)

E. (1, 3, 4, 5, 6, 0, 1)

5. Se dă următorul algoritm (vezi Algoritmul 5).

Algorithm 5 Algoritmul Ordonare prin insertie directa

```
1: procedure SORTARE_INSERTARE( $a, n$ )
2:   for  $i \leftarrow 2, n$  do
3:      $x \leftarrow a_i$ 
4:      $j \leftarrow i - 1$ 
5:     while  $(j > 0) \wedge (a_j > x)$  do
6:        $a_{j+1} \leftarrow a_j$ 
7:        $j \leftarrow j - 1$ 
8:     end while
9:      $a_{j+1} \leftarrow x$ 
10:  end for
11: end procedure
```

Care vor fi valorile vectorului a după terminarea pasului $i = 5$, știind că la intrare avem valorile $n = 7$ și $a = (4, 7, 2, 5, 6, 1, 3)$?

A. (4, 7, 2, 5, 6, 1, 3)

B. (2, 4, 7, 5, 6, 1, 3)

C. (2, 4, 5, 7, 6, 1, 3)

D. (2, 4, 5, 6, 7, 1, 3)

E. (1, 2, 4, 5, 6, 7, 3)

6. Se dă următorul algoritm (vezi Algoritmul 6).

Algorithm 6 Algoritmul de ordonare prin metoda shellsort

```
1: procedure SHELLSORT( $a, n$ )
2:    $step \leftarrow \lfloor \frac{n}{2} \rfloor$ 
3:    $k \leftarrow 1$ 
4:   while  $(step > 0) \wedge (k < 2)$  do
5:     for  $i \leftarrow step + 1, n$  do
6:        $j \leftarrow i - step$ 
7:       while  $(j > 0)$  do
8:         if  $(a_j > a_{j+step})$  then
9:            $a_j \leftrightarrow a_{j+step}$ 
10:           $j \leftarrow j - step$ 
11:        else
12:           $j \leftarrow 0$ 
13:        end if
14:      end while
15:    end for
16:     $step \leftarrow \lfloor \frac{step}{2} \rfloor$ 
17:     $k \leftarrow k + 1$ 
18:  end while
19: end procedure
```

Care vor fi valorile vectorului a la sfârșitul algoritmului, știind că la intrare avem valorile $n = 7$ și $a = (4, 7, 2, 5, 6, 1, 3)$?

A. (4, 1, 2, 5, 6, 7, 3)

B. (1, 2, 3, 4, 5, 6, 7)

C. (2, 3, 1, 5, 4, 7, 6)

D. (3, 6, 1, 4, 7, 2, 5)

E. (2, 1, 3, 4, 5, 7, 6)

7. Se dă următorul algoritm (vezi Algoritmul 7).

Algorithm 7 Algoritmul de ordonare prin interclasare

```
1: procedure MERGESORT( $A, n; B$ )
2:   if ( $n = 1$ ) then
3:      $b \leftarrow a$ 
4:   else
5:     if ( $n = 2$ ) then
6:       if ( $a_1 > a_2$ ) then
7:          $a_1 \leftrightarrow a_2$ 
8:       end if
9:        $b \leftarrow a$ 
10:    else
11:      $k \leftarrow k + 1$ 
12:      $middle \leftarrow \lfloor \frac{n}{2} \rfloor$ 
13:     call Split( $A, n, middle; A1, p, A2, q$ )
14:     call MergeSort( $A1, p; B1$ )
15:     call MergeSort( $A2, q; B2$ )
16:     call Merge( $B1, p, B2, q; B, n$ )
17:   end if
18: end if
19: end procedure
```

Care vor fi valorile vectorului a în momentul în care valoarea variabilei k devine 4 (linia 11), știind că la intrare avem valorile $n = 7$, $k = 1$ și $a = (6, 7, 5, 2, 4, 1, 3)$?

A. (6, 7, 5, 2, 4, 1, 3)

B. (6, 7, 2, 5, 4, 1, 3)

C. (2, 5, 6, 7, 4, 1, 3)

D. (2, 5, 6, 7, 1, 4, 3)

E. (2, 5, 6, 7, 1, 3, 4)

8. Se dă următorul algoritm (vezi Algoritmul 8).

Algorithm 8 Algoritmul Calculi

```
1: function CALCUL1( $n$ )
2:   if ( $n = 0$ ) then
3:      $v \leftarrow 1$ 
4:   else
5:     if ( $n \bmod 2 = 1$ ) then
6:        $v \leftarrow 2 \cdot calcul1(\lfloor \frac{n}{2} \rfloor)$ 
7:     else
8:        $v \leftarrow n + 1$ 
9:     end if
10:  end if
11:  return  $v$ 
12: end function
```

Ce valoare returneaza *Calcul1*(15)?

A. 15

B. 11

C. 13

D. 16

E. 14

9. Se dă următorul algoritm (vezi Algoritmul 9).

Algorithm 9 Algoritmul Calcul2

```
1: function CALCUL2( $x, y$ )
2:   if ( $y = 0$ ) then
3:      $v \leftarrow 0$ 
4:   else
5:     if ( $y \bmod 2 = 1$ ) then
6:        $v \leftarrow \text{calcul2}(2 \cdot x, \lfloor \frac{y}{2} \rfloor) + x$ 
7:     else
8:        $v \leftarrow \text{calcul2}(2 \cdot x, \lfloor \frac{y}{2} \rfloor)$ 
9:     end if
10:  end if
11:  return  $v$ 
12: end function
```

Ce valoare returneaza $\text{Calcul2}(7, 5)$?

A. 32

B. 30

C. 37

D. 35

E. 34

10. Se dă următorul algoritm (vezi Algoritmul 10).

Algorithm 10 Algoritmul interclasare optima

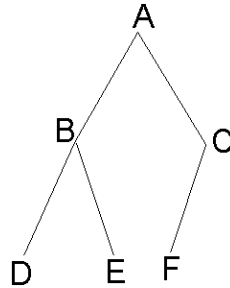
```
1: procedure INTERCLASAREOPTIMA( $n, L; ST, DR, C$ )
Require:    $\begin{cases} n & \text{numarul de siruri} \\ L & \text{vectorul lungimilor} \end{cases}$ 
2:    $C \leftarrow 0$ 
3:   for  $i \leftarrow 1, n$  do
4:      $P_i \leftarrow 0$ 
5:      $ST_i \leftarrow 0$ 
6:      $DR_i \leftarrow 0$ 
7:   end for
8:   for  $i \leftarrow 1, n - 1$  do
9:     call  $\text{MinLength}(n + i - 1, L, P, k)$ 
10:     $ST_{n+i} \leftarrow k$ 
11:     $P_k \leftarrow 1$ 
12:    call  $\text{MinLength}(n + i - 1, L, P, k)$ 
13:     $DR_{n+i} \leftarrow k$ 
14:     $P_k \leftarrow 1$ 
15:     $L_{n+i} \leftarrow L_{ST_i} + L_{DR_i}$ 
16:     $P_{n+i} \leftarrow 0$ 
17:     $C \leftarrow C + L_{n+i}$ 
18:  end for
19: end procedure
```

Care va fi valoarea variabile C după terminarea pasului $i = 4$, știind că la intrare avem valorile $n = 7$ și $L = (100, 80, 50, 100, 70, 110, 120)$?

A. 1610

- B. 680
- C. 1020
- D. 1030**
- E. 1040

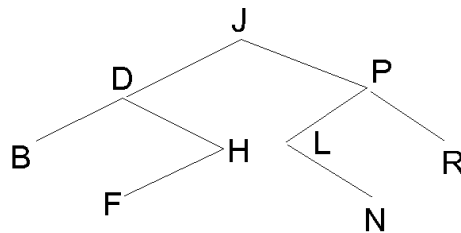
11. O procedură ce parcurge arborele binar



în *postordine* va afișa:

- A. A B C D E F
- B. A B D E C F
- C. D B E A F C
- D. D E B F C A**
- E. D E F B C A

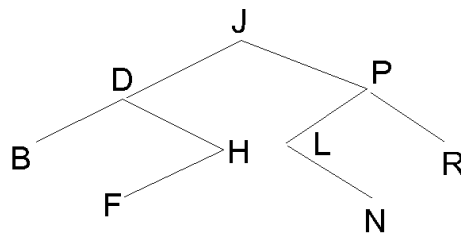
12. O procedură ce parcurge arborele binar



în *preordine* va afișa:

- A. J D P B H L R F N
- B. J D B H F P L N R**
- C. N F R L H B P D J
- D. B D H F N L P R J
- E. B D F H L J N P R

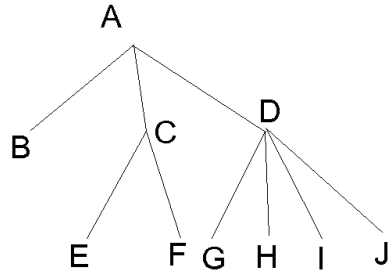
13. O procedură ce parcurge arborele binar



în *inordine* va afișa:

- A. B H F D L N P R J
- B. B D H F N L P R J
- C. N F R L H B P D J
- D. J D P B H L R F N
- E. B D F H J L N P R**

14. O procedură ce parcurge arborele binar



în *A*-preordine va afișa:

- A. A B C E F D G H I J**
- B. A B C D E F G H I J
- C. B E F C G H I J D A
- D. B A E C F G D H I J
- E. B E F G H I J C D A

15. Se dă următorul algoritm (vezi Algoritmul 11).

Algorithm 11 Algoritm Ordonare

```

1:  $k \leftarrow 0$ 
2: repeat
3:    $ok \leftarrow 0$ 
4:   for  $i \leftarrow 1, n - 1$  do
5:     if  $(x_i > x_{i+1})$  then
6:        $x_i \leftrightarrow x_{i+1}$ 
7:        $ok \leftarrow 1$ 
8:     end if
9:   end for
10:   $k \leftarrow k + 1$ 
11: until  $((ok \neq 1) \vee (k = 2))$ 
  
```

Știind că datele de intrare sunt $n = 7$ și $x = 7, 6, 5, 4, 3, 2, 1$ care vor fi valorile vectorului x la sfârșitul algoritmului?

- A. 1, 2, 3, 4, 5, 6, 7
- B. 6, 7, 3, 4, 5, 2, 1
- C. 5, 4, 3, 2, 1, 6, 7**
- D. 4, 3, 2, 1, 5, 6, 7
- E. 2, 1, 4, 3, 6, 5, 7

16. Se dă următorul algoritm (vezi Algoritmul 12).

Algorithm 12 Algoritm Ordonare

```
1:  $k \leftarrow 0$ 
2: repeat
3:    $ok \leftarrow 0$ 
4:   for  $i \leftarrow 1, n - 1$  do
5:     if  $(x_i > x_{i+1})$  then
6:        $x_i \leftrightarrow x_{i+1}$ 
7:        $ok \leftarrow 1$ 
8:     end if
9:   end for
10:   $k \leftarrow k + 1$ 
11: until  $((ok \neq 1) \vee (k = 2))$ 
```

Știind că datele de intrare sunt $n = 7$ și $x = 7, 5, 3, 1, 2, 4, 6$ care vor fi valorile vectorului x la sfârșitul algoritmului?

- A. 3, 1, 2, 4, 5, 6, 7
- B. 6, 7, 3, 4, 5, 2, 1
- C. 5, 4, 3, 2, 1, 6, 7
- D. 4, 3, 2, 1, 5, 6, 7
- E. 2, 1, 4, 3, 6, 5, 7

17. Se dă următorul algoritm (vezi Algoritmul 13).

Algorithm 13 Algoritm Ordonare

```
1:  $k \leftarrow 0$ 
2: repeat
3:    $ok \leftarrow 0$ 
4:   for  $i \leftarrow 1, n - 1$  do
5:     if  $(x_i > x_{i+1})$  then
6:        $x_i \leftrightarrow x_{i+1}$ 
7:        $ok \leftarrow 1$ 
8:     end if
9:   end for
10:   $k \leftarrow k + 1$ 
11: until  $((ok \neq 1) \vee (k = 2))$ 
```

Știind că datele de intrare sunt $n = 7$ și $x = 1, 2, 3, 7, 6, 5, 4$ care vor fi valorile vectorului x la sfârșitul algoritmului?

- A. 1, 2, 3, 6, 7, 4, 5
- B. 1, 2, 3, 4, 5, 6, 7
- C. 1, 2, 3, 5, 4, 6, 7
- D. 1, 2, 3, 6, 5, 4, 7
- E. 1, 2, 3, 7, 6, 5, 4

18. Se dă următorul algoritm (vezi Algoritmul 14).

Știind că datele de intrare sunt $n = 7$ și $x = 2, 1, 7, 6, 5, 4, 3$ care vor fi valorile vectorului x la sfârșitul algoritmului?

- A. 1, 2, 3, 4, 5, 6, 7
- B. 6, 7, 3, 4, 5, 2, 1

Algorithm 14 Algoritm Ordonare

```
1:  $k \leftarrow 0$ 
2: repeat
3:    $ok \leftarrow 0$ 
4:   for  $i \leftarrow 1, n - 1$  do
5:     if  $(x_i > x_{i+1})$  then
6:        $x_i \leftrightarrow x_{i+1}$ 
7:        $ok \leftarrow 1$ 
8:     end if
9:   end for
10:   $k \leftarrow k + 1$ 
11: until  $((ok \neq 1) \vee (k = 2))$ 
```

C. 5, 4, 3, 2, 1, 6, 7

D. 1, 2, 5, 4, 3, 6, 7

E. 2, 1, 4, 3, 6, 5, 7

19. Se dă următorul algoritm (vezi Algoritmul 15).

Algorithm 15 Algoritm Ordonare

```
1:  $k \leftarrow 0$ 
2: repeat
3:    $ok \leftarrow 0$ 
4:   for  $i \leftarrow 1, n - 1$  do
5:     if  $(x_i > x_{i+1})$  then
6:        $x_i \leftrightarrow x_{i+1}$ 
7:        $ok \leftarrow 1$ 
8:     end if
9:   end for
10:   $k \leftarrow k + 1$ 
11: until  $((ok \neq 1) \vee (k = 2))$ 
```

Știind că datele de intrare sunt $n = 7$ și $x = 2, 7, 3, 5, 6, 4, 1$ care vor fi valorile vectorului x la sfârșitul algoritmului?

A. 2, 1, 3, 5, 6, 4, 7

B. 6, 7, 3, 4, 5, 2, 1

C. 5, 4, 3, 2, 1, 6, 7

D. 2, 3, 5, 4, 1, 6, 7

E. 1, 2, 3, 4, 5, 6, 7

20. Se dă un vector cu n componente la început nule. Pentru $1 \leq i < \left[\frac{i+j}{2}\right] < j \leq n$ definim "secționarea" vectorului corespunzător perechii de indici (i, j) astfel:
- Pas 1. Incrementarea cu p unități a elementelor aflate între pozițiile i și k .
- Pas 2. "Secționarea" corespunzătoare perechii (i, k) .
- Pas 3. Incrementarea cu q unități a elementelor aflate între pozițiile k și j .
- Pas 4. "Secționarea" corespunzătoare perechii (k, j) . unde $k = \left[\frac{i+j}{2}\right]$.
- Pentru $n = 7, p = 1, q = 2$ să se indice care este vectorul rezultat în urma secționării corespunzătoare perechii $(1, n)$.
- A. (2, 5, 6, 7, 5, 7, 6)
- B. (1, 5, 6, 8, 6, 7, 5)
- C. (2, 5, 6, 8, 6, 7, 6)**
- D. (2, 5, 7, 8, 6, 7, 7)
- E. (2, 4, 6, 7, 5, 6, 5)
21. Se dă un vector cu n componente la început nule. Pentru $1 \leq i < \left[\frac{i+j}{2}\right] < j \leq n$ definim "secționarea" vectorului corespunzător perechii de indici (i, j) astfel:
- Pas 1. Incrementarea cu p unități a elementelor aflate între pozițiile i și k .
- Pas 2. "Secționarea" corespunzătoare perechii (i, k) .
- Pas 3. Incrementarea cu q unități a elementelor aflate între pozițiile k și j .
- Pas 4. "Secționarea" corespunzătoare perechii (k, j) . unde $k = \left[\frac{i+j}{2}\right]$.
- Pentru $n = 6, p = 3, q = 1$ să se indice care este vectorul rezultat în urma secționării corespunzătoare perechii $(1, n)$.
- A. (6, 7, 8, 8, 6, 3)**
- B. (5, 6, 7, 7, 5, 2)
- C. (6, 7, 6, 6, 7, 4)
- D. (6, 7, 6, 6, 5, 3)
- E. (6, 7, 8, 8, 5, 4)
22. Se dă un vector cu n componente la început nule. Pentru $1 \leq i < \left[\frac{i+j}{2}\right] < j \leq n$ definim "secționarea" vectorului corespunzător perechii de indici (i, j) astfel:
- Pas 1. Incrementarea cu p unități a elementelor aflate între pozițiile i și k .
- Pas 2. "Secționarea" corespunzătoare perechii (i, k) .
- Pas 3. Incrementarea cu q unități a elementelor aflate între pozițiile k și j .
- Pas 4. "Secționarea" corespunzătoare perechii (k, j) . unde $k = \left[\frac{i+j}{2}\right]$.
- Pentru $n = 6, p = 1, q = 2$ să se indice care este vectorul rezultat în urma secționării corespunzătoare perechii $(1, n)$.
- A. (2, 4, 5, 5, 7, 6)
- B. (2, 4, 6, 6, 7, 6)**
- C. (1, 4, 6, 7, 6, 5)
- D. (2, 5, 6, 6, 8, 7)
- E. (2, 4, 5, 5, 6, 6)

23. Se dă un vector cu n componente la început nule. Pentru $1 \leq i < \lfloor \frac{i+j}{2} \rfloor < j \leq n$ definim "secționarea" vectorului corespunzător perechii de indici (i, j) astfel:

Pas 1. Incrementarea cu p unități a elementelor aflate între pozițiile i și k .

Pas 2. "Secționarea" corespunzătoare perechii (i, k) .

Pas 3. Incrementarea cu q unități a elementelor aflate între pozițiile k și j .

Pas 4. "Secționarea" corespunzătoare perechii (k, j) . unde $k = \lfloor \frac{i+j}{2} \rfloor$.

Pentru $n = 7, p = 3, q = 1$ să se indice care este vectorul rezultat în urma secționării corespunzătoare perechii $(1, n)$.

A. (6, 8, 8, 9, 8, 6, 4)

B. (6, 10, 9, 10, 8, 6, 3)

C. (6, 10, 8, 9, 7, 6, 4)

D. (6, 10, 8, 9, 8, 6, 3)

E. (5, 9, 8, 9, 8, 7, 3)

24. Se dă un vector cu n componente la început nule. Pentru $1 \leq i < \lfloor \frac{i+j}{2} \rfloor < j \leq n$ definim "secționarea" vectorului corespunzător perechii de indici (i, j) astfel:

Pas 1. Incrementarea cu p unități a elementelor aflate între pozițiile i și k .

Pas 2. "Secționarea" corespunzătoare perechii (i, k) .

Pas 3. Incrementarea cu q unități a elementelor aflate între pozițiile k și j .

Pas 4. "Secționarea" corespunzătoare perechii (k, j) . unde $k = \lfloor \frac{i+j}{2} \rfloor$.

Pentru $n = 7, p = 2, q = 3$ să se indice care este vectorul rezultat în urma secționării corespunzătoare perechii $(1, n)$.

A. (4, 8, 9, 12, 9, 10, 8)

B. (4, 7, 9, 11, 9, 8, 7)

C. (6, 10, 11, 14, 11, 13, 10)

D. (4, 10, 11, 15, 11, 12, 10)

E. (4, 9, 10, 13, 10, 11, 9)

25. Se dă următorul algoritm (vezi Algoritmul 16).

Algorithm 16 Algoritm recursiv

```
1: function F(n)
2:   if (n mod 2 = 0) then
3:     v ← n
4:   else
5:     v ← 0
6:   end if
7:   return v
8: end function
9: function S(a, b)
10:  if (a > b) then
11:    v ← 0
12:  else
13:    v ← f(a) + S(a + 1, b)
14:  end if
15:  return v
16: end function
```

Care va fi rezultatul apelului funcției $S(2, 10)$?

A. 24

- B. 28
- C. 31
- D. 30**
- E. 34

26. Se dă următorul algoritm (vezi Algoritmul 17).

Algorithm 17 Algoritm recursiv

```
1: function F(n)
2:   if (n mod 2 = 0) then
3:     v ← n
4:   else
5:     v ← 0
6:   end if
7:   return v
8: end function
9: function S(a, b)
10:  if (a > b) then
11:    v ← 0
12:  else
13:    v ← f(a) + S(a + 1, b)
14:  end if
15:  return v
16: end function
```

Care va fi rezultatul apelului funcției $S(3, 11)$?

- A. 26
- B. 30
- C. 31
- D. 28**
- E. 34

27. Se dă următorul algoritm (vezi Algoritmul 18).

Algorithm 18 Algoritm recursiv

```
1: function F(n)
2:   if (n mod 2 = 0) then
3:     v ← n
4:   else
5:     v ← 0
6:   end if
7:   return v
8: end function
9: function S(a, b)
10:  if (a > b) then
11:    v ← 0
12:  else
13:    v ← f(a) + S(a + 1, b)
14:  end if
15:  return v
16: end function
```

Care va fi rezultatul apelului funcției $S(3, 15)$?

- A. 79

- B. 45
- C. 52
- D. 54**
- E. 57

28. Se dă următorul algoritm (vezi Algoritmul 19).

Algorithm 19 Algoritm recursiv

```
1: function S(x, n)
2:   if (n > 0) then
3:     if (n mod 2 = 0) then
4:       y ← x + 10
5:     else
6:       y ← x + 20
7:     end if
8:     v ← x + S(y, n - 1)
9:   else
10:    v ← 0
11:  end if
12:  return v
13: end function
```

Care va fi rezultatul apelului funcției $S(500, 4)$?

- A. 1550
- B. 1160
- C. 2070
- D. 2080**
- E. 2150

29. Se dă următorul algoritm (vezi Algoritmul 20).

Algorithm 20 Algoritm recursiv

```
1: function S(x, n)
2:   if (n > 0) then
3:     if (n mod 2 = 0) then
4:       y ← x + 10
5:     else
6:       y ← x + 20
7:     end if
8:     v ← x + S(y, n - 1)
9:   else
10:    v ← 0
11:  end if
12:  return v
13: end function
```

Care va fi rezultatul apelului funcției $S(200, 5)$?

- A. 1550
- B. 2080
- C. 2070
- D. 1160**

Algorithm 21 Algoritm recursiv

```
1: function S(x, n)
2:   if (n > 0) then
3:     if (n mod 2 = 0) then
4:       y ← x + 10
5:     else
6:       y ← x + 20
7:     end if
8:     v ← x + S(y, n - 1)
9:   else
10:    v ← 0
11:  end if
12:  return v
13: end function
```

E. 2150

30. Se dă următorul algoritm (vezi Algoritmul 21).

Care va fi rezultatul apelului funcției $S(100, 6)$?

A. 1010

B. 1160

C. 1140

D. 810

E. 840

2 Programare procedurala

1. Ce se va afișa pe ecran în urma execuției următorului program?

```
#include <stdio.h>
void main(){
    int a,b = 0;
    int i;
    for (i=0;i<10;i++){
        b = a;
        a = i;
    }
    printf("%d", b);
}
```

A. 8

B. 9

C. 10

D. 0123456789

2. Care din următoarele expresii sunt echivalente cu `a[i]`?

A. `*(a+i)`

B. `i[a]`

C. `&a[i]`

D. `*(a+a)`

E. `*(a+i*sizeof(a[i]))`

3. Instrucțiunea `k++` este echivalentă cu:

A. `k = k + 1`

B. `k += 1`

C. `++k`

D. `k = k + k`

E. `k += k`

F. `k = k + sizeof(k)`

4. Ce se va afișa pe ecran în urma execuției următorului program?

```
#include <stdio.h>
int a,b;
void f1 (int *r, int *s){
    int temp;
    temp = *r;
    *r = *s;
    *s = temp;
}
void f2 (int *x, int *y){
    if (*x > *y) f1(x,y);
}
main(){
    a = 64;
    b = 42;
    f2(&a,&b);
    printf("%d,%d\n",a,b);
}
```


A. 42,64

B. 64,42

C. 64,64

D. 42,42

E. eroare de compilare pentru că lipsește void la declararea funcției main

5. Ce se va afișa pe ecran în urma execuției următorului program?

```
#include <stdio.h>
int a,b;
void f1 (int r, int s){
    int temp;
    temp = r;
    r = s;
    s = temp;
}
void f2 (int x, int y){
    if (x > y) f1(x,y);
}
void main(){
    a = 064;
    b = 042;
    f2(a,b);
    printf("%d,%d\n",a,b);
}
```

A. 52,34

B. 64,42

C. 42,62

D. 34,52

E. 0,0

F. rezultatul nu poate fi determinat având în vedere valorile variabilelor a și b

6. Ce se va afișa pe ecran în urma execuției următorului program?

```
#include <stdio.h>
void main(){
    int k[ ] = {100,200,300,400,500,600,700};
    int *t=k+2;
    printf("%d ",*t);
    printf("%d ",*(t+2) + *t);
    *t = *t + 10;
    printf("%d ",*t);
    t = t+3;
    printf("%d ",*t);
    *t = *t + *(t-2);
    printf("%d ",*t);
}
```

A. 300 800 310 600 1000

B. 200 300 110 600 1100

C. 300 900 320 600 1000

- D. 300 800 310 600 900
- E. 100 900 310 600 1000
- F. nici una din variante nu este corecta

7. Ce se va afișa pe ecran în urma execuției următorului program?

```
#include <stdio.h>
void main(){
    int p[] = {50,60,70,80,90,100};
    int *q[6],i;
    for(i =0;i<6;i++)
        q[i] = &p[i];

    for(i=5;i>=1;i--){
        q[i] = q[i-1];
        *q[i] = *q[i] + *p * 2;
    }
    for(i=0;i<6;i++)
        printf("%d ",p[i]);
}
```

A. 150 160 170 180 190 100

B. 160 160 170 180 190 100

C. 150 150 150 150 150 150

D. 150 160 170 180 190 190

E. 150 160 170 180 190 200

F. nici una din variante nu este corectă

8. Ce se va afișa pe ecran în urma execuției următorului program?

```
#include <stdio.h>
#include <stdlib.h>
void f(int *x){
    x=(int *) malloc(sizeof(int));
    *x=12;
}
void main(){
    int v=15/10;
    f(&v);
    printf("%d",v);
}
```

A. 1

B. 12

C. 15

D. 1.5

E. adresa variabilei v

F. adresa variabilei x

9. Ce se va afișa pe ecran în urma execuției următorului program?

```

#include <stdio.h>
void main(){
    int i, p=0x10;
    for(i=1;;i+=2){
        if (i=5)
            break;
        p+=i;
    }
    printf("%d",p);
}

```

A. 16

B. 10

C. 26

D. 25

E. 14

F. programul ciclează la infinit

10. Ce se va afișa pe ecran în urma execuției următorului program?

```

#include <stdio.h>
int a = 10, b = 5;
void f (int a, int t[]){
    b = a;
    t[2] = a % 2;
    a = (t[0]+t[1])/2;
    t[0]^=t[1];
    t[1]^=t[0];
    t[0]^=t[1];
}
void main(){
    int t[]={4,3,1};
    f(t[2],t);
    printf("%d %d %d %d %d",a,b,t[0],t[1],t[2]);
}

```

A. 10 1 3 4 1

B. 10 5 4 3 1

C. 1 5 3 4 1

D. 7 5 3 4 1

E. 5 5 10 4 0

F. nici una din variante nu este corectă

11. Considerăm următoarea secvență de program?

```

float x = 10;
int y = 7;
float t[]={3,4,1};
float *q = &x;
int *r = &y;
void *p;

```

Care din următoarele atribuiri sunt corecte?

- A. q = t;
- B. p = t;
- C. p = &y;
- D. p = r;
- E. x = &t[2];
- F. t = q;
- G. q = r;
- H. r = q;
- I. r = p;

12. Ce se va afișa pe ecran în urma execuției următorului program?

```
#include <stdio.h>
void main(){
  char s[]="abcdef";
  char *p,*q,c;
  p=s;
  q=p+4;
  while(p < q){
    c=*p;
    *p=*q;
    *q=++c;
    p++;q--;
  }
  printf("%s",s);
}
```

- A. edccbf
- B. fedcba
- C. feddca
- D. edcbaf
- E. bcdefg
- F. nici una din variante nu este corectă

13. Care din următoarele linii vor genera eroare la compilare?

```
1: void main() {
2:   int a[2] = {1,2};
3:   int b[3] = {3};
4:   int* x = a;
5:   int* const y = a;
6:   b = x;
7:   b = y;
8:}
```

- A. 6, 7
- B. 3, 5, 6, 7
- C. 3, 4, 6, 7
- D. 3, 5

E. 5, 6, 7

F. 5

G. nici una

14. Considerăm următorul program:

```
1: #include <stdio.h>
2: void main(){
3:  int t[10];
4:  printf("sizeof(int)=%d\n",sizeof(int));
5:  printf("%p\n",t);
6:  printf("%p\n",t+3);
7:}
```

Dacă în urma execuției liniilor 4 și 5 se afișează 2 respectiv 0xF004, ce se va afișa în urma execuției liniei 6?

A. 0xF00A

B. 0xF010

C. 0xF007

D. 0xF006

E. 0xF003

15. Ce se va afișa pe ecran în urma execuției următorului program?

```
#include <stdio.h>
void main(){
  int x=7;
  x = 1,2,3;
  printf("%d",x);
}
```

A. 1

B. 7

C. 2

D. 3

E. 6

F. nimic (programul nu se compilează)

Solution: Operatorul virgulă se evaluează de la stânga la dreapta. Rezultatul este dat de ultima valoare a expresiei. Însă, deoarece operatorul = are prioritatea mai mare decât a operatorului virgulă mai întâi se atribuie lui x valoarea 1 și apoi se evaluează expresia x,2,3, rezultat care nu este memorat în nicio variabilă.

16. Ce se va afișa pe ecran în urma execuției următorului program?

```
#include <stdio.h>
void main() {
  int array[3][3] = {{0, 1, 2}, {3, 4, 5}, {6, 7, 8}};
  int s = 0, i ,j;
  for (i = 0; i < 3 ; ++i){
    for (j = 2; j < 3 ; j++){
```

```

        s += array[i][j];
    }
}
printf("%d",s);
}

```

A. nimic (programul nu se compilează)

B. 15

C. 0

D. 27

E. 36

Solution: La declararea tablourilor multidimensionale numai prima dimensiune poate fi omisă. În acest caz, se generează eroare de compilare deoarece nu este specificat numărul de coloane.

17. Ce se va afișa pe ecran în urma execuției următorului program?

```

#include <stdio.h>

int f(int y);

void main(){
    int x = 3;
    int y = 6;
    printf("%d", f(x));
}

int f(int x){
    return x+1;
}

```

A. 4

B. 7

C. 3

D. 6

E. nimic (eroare de compilare deoarece funcția **f** a fost apelată înainte de a fi definită)

F. nimic (eroare de compilare deoarece numele argumentului funcției **f** este **y** la declarare și **x** la definire)

18. Câte elemente ale vectorului **a** vor avea valoarea 9 după execuția programului de mai jos?:

```

void main(){
    int a[] = {0, 1, 2, 3, 0, 4, 5, 6};
    int i = 0, x = 9;

    do{
        a[i++] = x;
    }while(i<6&& a[i]);
}

```

- A. nici unul
- B. unu
- C. două
- D. patru**
- E. toate

19. Fie urmatorul program:

```
void main (){
    int v[20], i, n, D;

    scanf("%d", &n);
    for(i=0;i<n;i++)
        v[i]=i%2?-i;
    for(D=1,i=0;i<n;D*=v[i++]);
        D++;
    printf("%d",D);
}
```

În urma execuției sale sunt posibile următoarele situații:

- A. Expresia condițională din primul ciclu for este eronată din punct de vedere sintactic.
- B. Dacă variabila n primește la citire valoarea 6, atunci elementele vectorului v vor fi, în ordine (0,1,-2,3,-4,5).**
- C. Prezența caracterului ” ; ” după al doilea ciclu for constituie o eroare
- D. Dacă variabila n primește la citire valoarea 5, atunci programul afișează 1**
- E. Programul funcționează corect pentru orice valoare întregă a lui n mai mică sau egală cu MAXINT.

20. Fie programul :

```
void main(){
    int v[]={0, 1, 2, 3, 4, 5, 0};
    int i=0, n=0;

    do{
        if (i == v[i])
            n++;
    }while(i<6 && v[i++]);
}
```

În urma execuției programului sunt posibile următoarele situații:

- A. variabila n va avea valoarea 0
- B. variabila n va avea valoarea 1**
- C. programul va intra într-un ciclu infinit
- D. variabila n va avea valoarea 5
- E. variabila n va avea valoarea 2

21. Se consideră secvența următoare, în care valorile lui n și x se presupun cunoscute, v este un vector cu elementele(v[0],v[1],...,v[n-1])

```

p=n;
for(i=0;i<n;i++)
    if (v[i]==x)
        p=i;
for(i=p+1; i<n; i++)
    v[i-1]=v[i];
for(i=0; i<n-1; i++)
    printf("%3d",v[i]);

```

Precizați care dintre următoarele afirmații sunt adevărate:

- A. Pentru $n=5$, $x=3$ și $v=(5,6,2,7,1)$, se afișează ultimele patru elemente nemodificate ale vectorului: 6 2 7 1.
- B. Pentru $n=5$, $x=1$ și $v=(2,1,3,1,4)$, se afișează: 2 3 1 4;
- C. Secvența conține erori de sintaxă
- D. Algoritmul șterge din vector elementul cu valoarea x , prin mutarea cu o poziție mai la dreapta a elementelor aflate înaintea lui.

E. Algoritmul șterge din vector elementul cu valoarea x , prin mutarea cu o poziție mai la stânga a elementelor aflate după el.

22. În programul următor, care dintre secvențele de instrucțiuni (I), (II), (III) realizează corect citirea unui șir de caractere de la tastatură și afișarea acestuia?

```

void main(){
    char s1[10],s2[10],s3[10];
    scanf("%s", &s3[3]);
    printf("%s", s3[3]);           //(I)
    scanf("%s", s2);
    printf("s2=%s", s2);          //(II)
    scanf("%s",&s1);
    printf("%s", s1[10]);         //(III)
}

```

- A. numai (I)
- B. toate
- C. (I) și (II)
- D. (I) și (III)

E. numai (II)

23. Pentru programul următor, analizați corectitudinea afirmațiilor de mai jos:

```

#include <stdlib.h>
#include <string.h>
#include <stdio.h>

void main(){
    char s1[4], s2[4];
    long x;

    scanf("%s %s", s1, s2) ;
    if (strcmp(s1, s2)>0)
        x=atol(s1);

```



```

else
    if(strcmp(s1, s2)==0)
        x = 0;
    else x = atol(s2);
printf("%d", x);
}

```

A. Condițiile din cele doua linii if sunt greșite.

B. Apelurile funcției atol sunt corecte

C. Dacă de la tastatura se introduc șirurile "98" și "123" atunci se va afișa 98.

D. Dacă de la tastatura se introduc șirurile "123" și "121", atunci programul va afișa șirul "123".

E. Dacă de la tastatura se introduc șirurile "ab" și "ac", atunci se va semnala un mesaj de EROARE.

24. În conformitate cu standardul ASCII, codurile literelor mari sunt succesive începând cu 65, ce va afișa programul de mai jos?

```

#include <stdlib.h>
#include <string.h>
#include <stdio.h>

void main(){
    int x = 20, e;
    char s[15] ="ABC", t[15], u[15];

    e = s[1] + s[2];
    itoa(e, t, 10);
    strcpy(u, t);
    strcat(s, u);
    printf("%s ", s);
}

```

A. Nimic, șirul s fiind vid

B. ABC13

C. AB13

D. ABC133

E. ABC131

25. Care dintre specificatorii modului de acces al fișierelor binare sau text, pentru funcția fopen, sunt corecți:

A. ab

B. rb

C. ap

D. wt

E. at

F. ba

G. toate variantele sunt corecte

26. Știind că fișierul f1.txt există în directorul curent iar f2.txt nu este creat, care dintre următoarele secvențe de instrucțiuni este greșită?

- A. f1=fopen("f1.txt","w");
- B. f1=fopen("f1.txt","r+");
- C. f2=fopen("f2.txt","r+");**
- D. f2=fopen("f2.txt","w+");
- E. toate variantele anterioare sunt greșite

27. Care este efectul subprogramului alăturat?

```
void X(char *a, char *b){
    FILE *f,*g;
    char s[255]

    f=fopen(a,"a");
    g=fopen(b,"r");
    while(!feof(g)){
        fgets(s,255,g);
        fputs(s,f);
    }
    f.close(f);
    fclose(g);
}
```

- A. copiază conținutul fișierului g peste conținutul fișierului f
- B. citește informațiile din cele două fișiere ale căror nume se transmit ca parametri
- C. concatenează două fișiere, rezultatul concatenării fiind pus în fișierul f**
- D. concatenează două fișiere, rezultatul concatenării fiind pus în fișierul g
- E. toate variantele anterioare sunt greșite

28. Care dintre următoarele afirmații sunt adevărate?

- A. Pentru a închide un fișier se folosește funcția unlink;
- B. Pentru redenumirea unui fișier în cadrul programului se folosește funcția remove
- C. Pentru a deschide un fișier se folosește funcția fopen**
- D. Nu pot fi adăugate informații într-un fișier
- E. Toate celelalte variante sunt eronate

29. Ce valori se vor găsi în fișierul numere.txt după execuția următorului program?

```

#include <conio.h>
#include <stdio.h>

FILE *f;
int i=0, a[10]={20,11,17,4,5,10,14,34,23,11};

void main(){
    f=fopen("c:\\numere.txt","w");
    for(i=0; i<5; i++){
        if(a[i]%2!=0)
            fprintf(f,"%d \n",a[i]);
    }
    fclose(f);
}

```

A. 11 17 5 23 11

B. 20 4 10 14 34

**C. 11
17
5**

D. 11
17
5
23
11

30. Ce se întâmplă în urma execuției următorului program dacă fișierul nr.txt conține valorile 7 14 6 3 8 10 ?

```

FILE *f, *g;
int x;
void main(){
    f=fopen("nr.txt","r+");
    g=fopen("nr2.txt","w+");
    while(!feof(f)){
        fscanf(f,"%d", &x);
        if(x%2==0)
            fprintf(g,"%d \n",x);
    }
    fclose(f);
    fclose(g);
}

```

A. Fișierul nr2.txt va conține valorile 14 6 8 10

B. Fișierul nr2.txt va conține valorile 14 6 8 10, fiecare pe altă linie

C. Fișierul nr.txt va conține valorile 7 14 6 8 10

D. Fișierul nr.txt va conține valorile 7 14 6 8 10, fiecare pe altă linie

3 Programare orientata pe obiecte(C++)

1. Un program orientat obiect reprezintă:
 - A. o colecție de obiecte care cooperează prin intermediul mesajelor în vederea realizării unui obiectiv comun**
 - B. o variantă de program procedural
 - C. un program care poate fi considerat un obiect de către alte programe
 - D. un program specific mediului de programare Windows
 - E. un program format din mai multe fișiere sursa și header
2. O clasă în modelul de programare orientat obiect reprezintă:
 - A. o colecție de obiecte cu aceleași caracteristici și cu un comportament comun**
 - B. o colecție de caracteristici
 - C. un obiect din program
 - D. o interfață
 - E. un grup de comportamente specific unui obiect
3. Obiectele aparținând unei aceleiași clase se diferențiază prin:
 - A. starea lor**
 - B. valorile datelor membru declarate în cadrul clasei**
 - C. valorile caracteristicilor clasei**
 - D. caracteristicile diferite specifice fiecărui obiect
 - E. comportamentul diferit specific fiecărui obiect
4. Considerăm următorul program C++

```
#include <iostream>
using namespace std;
class C
{
private:
int _i;
public:
    C(int i):_i(i){}
    C():_i(3){}
    ~C(){cout<<_i<<endl;}
};
void main()
{
    C c1(4);
    C c2;
}
```

Care din următoarele afirmații sunt adevărate:

- A. programul nu conține erori de sintaxă**
- B. există erori de sintaxă în definițiile metodelor constructor
- C. destructorul nu poate conține instrucțiuni de afișare
- D. în cadrul programului principal sunt create două instanțe ale clasei C**

E. programul afișează la consolă numerele 3 și 4

5. Care din următoarele afirmații referitoare la moștenire sunt adevărate relativ la limbajul de programare C++:

A. Relația de moștenire în care clasa de bază și clasa derivată au aceeași interfață se numește moștenire pură

B. Dintr-o clasă de bază se poate obține prin intermediul moștenirii cel mult o clasă derivată

C. O clasă derivată are exact o clasă de bază

D. Obiectele aparținând unei clase derivate pot fi convertite la tipul unei clase de bază din care clasa derivată provine

E. Specificatorul protected este folosit pentru a arăta că elementele care îl urmează pot fi accesate și din cadrul claselor derivate

6. Ce se va afișa pe ecran în urma execuției următorului program:

```
#include <iostream.h>
class Test {
    int n;
public:
    Test(int x):n(x){}
    int f(int n, int &x, int *p){
        n=this->n;
        x=n++;
        *p=this->n + n;
        return n;
    }
};
void main(){
    Test t(2);
    int n=6,x=0x10,p=1;
    cout<<t.f(n,x,&p)<<" ";
    cout<<n<<" "<<x<<" "<<p;
}
```

A. 3 6 2 5

B. 3 6 16 1

C. 6 6 10 1

D. 3 7 10 1

E. Eroare de compilare deoarece constructorul nu conține nici o instrucțiune

7. Ce se va afișa pe ecran în urma execuției următorului program:

```
#include <iostream.h>
class B1 {
public:
    B1(){cout<<"B1 "};
    ~B1(){cout<<"DB1 "};
};
class B2 {
public:
    B2(){cout<<"B2 "};
    ~B2(){cout<<"DB2 "};
};
class D: public B1, public B2{
```

```

public:
    D():B2(),B1(){cout<<"D "};
    ~D(){cout<<"DD "};
};
void main(){
    D d1;
    D d2=d1;
}

```

A. B1 B2 D DD DB2 DB1 DD DB2 DB1

B. B2 B1 D DD DB2 DB1 DD DB2 DB1

C. B1 B2 D B1 B2 D DD DB2 DB1 DD DB2 DB1

D. B1 B2 D DD DB1 DB2 DD DB1 DB2

E. D DD

8. Ce se va afișa pe ecran în urma execuției următorului program:

```

#include <iostream.h>
class B1 {
public:
    B1(){cout<<"B1 "};
    ~B1(){cout<<"DB1 "};
};
class B2 {
public:
    B2(){cout<<"B2 "};
    ~B2(){cout<<"DB2 "};
};
class D: public B1, public B2{
    static int idGen;
    int id;
public:
    D():B2(),B1(){id=idGen++;cout<<"D("<<id<<") "};
    D(const D &o){id=idGen++;cout<<"CD("<<id<<") "};
    ~D(){cout<<"DD "};
};
int D::idGen = 0;

void main(){
    cout<<endl;
    D d1;
    D d2=d1;
}

```

A. B1 B2 D(0) B1 B2 CD(1) DD DB2 DB1 DD DB2 DB1

B. B1 B2 D(0) DD DB2 DB1 DD DB2 DB1

C. B1 B2 D(0) DD DB2 DB1

D. B2 B1 D(0) B1 B2 CD(1) DD DB2 DB1 DD DB2 DB1

E. B1 B2 D(0) B2 B1 CD(1) DD DB2 DB1 DD DB2 DB1

F. rezultatul executiei nu poate fi determinat deoarece depinde de continutul memoriei

9. Ce se va afișa pe ecran în urma execuției următorului program:

```

#include <iostream.h>
class B1 {
public:
    B1(){cout<<"B1 ";}
    ~B1(){cout<<"DB1 ";}
};
class B2 {
public:
    B2(){cout<<"B2 ";}
    ~B2(){cout<<"DB2 ";}
};
class D: public B1, public B2{
public:
    D():B2(),B1(){cout<<"D ";}
    D(const D &o){cout<<"CD ";}
    ~D(){cout<<"DD ";}
};
void main(){
    D *d1 = new D;
    D *d2 = d1;
    delete d1;
}

```

A. B1 B2 D DD DB2 DB1

B. B1 B2 D DD DB2 DB1 DD DB2 DB1

C. B2 B1 D DD DB2 DB1

D. B2 B1 D DD DB2 DB1 DD DB2 DB1

E. B1 B2 D B1 B2 D DD DB2 DB1 DD DB2 DB1

F. B1 B2 D B1 B2 D DD DB2 DB1

10. Ce se va afișa pe ecran în urma execuției următorului program:

```

#include <iostream.h>
class A {
public:
    A(){cout<<"A()";}
    A(int n,int m=0){cout<<"A(int,int)";}
    A(const A &o){cout<<"A(const A&)";}
    ~A(){cout<<"~A";}
};

void main(){
    cout<<endl;
    A a1;
    A a2=a1;
    A a3=1;
    A a4(1);
    A a5(1,1);
}

```

A. A()A(const A&)A(int,int)A(int,int)A(int,int)~A~A~A~A~A

B. A(const A&)A(const A&)A(int)A(int,int)A(int,int)~A~A~A~A~A

C. A()A(const A&)A(int,int)A(int,int)A(int,int)~A~A~A~A

D. A()A(const A&)A(int)A(int)A(int,int)~A~A~A~A~A

E. A()A()A(int,int)A(int)A(int,int)~A~A~A~A~A

F. Eroare de compilare la declararea obiectului a2 deoarece nu e supraîncărcat operatorul =.

11. Ce se va afișa pe ecran în urma execuției următorului program:

```
#include <iostream.h>
class B {
public:
    virtual void f() { cout<<"B::f() ";}
    void g() { cout<<"B::g() ";}
};
class D1: public B{
public:
    void f() { cout<<"D1::f() ";}
    void g() { cout<<"D1::g() ";}
};
class D2: public B{
public:
    void g() { cout<<"D2::g() ";}
};
void main(){
    int i;
    B *b[] = {new B(), new D1(), new D2()};
    for (i=2;i>=0;i--) {b[i]->f();}
    for (i=0;i<=2;i++) {b[i]->g();}
}
```

A. B::f() D1::f() B::f() B::g() B::g() B::g()

B. D2::f() D1::f() B::f() B::g() B::g() B::g()

C. B::f() D1::f() D::f() B::g() D1::g() D2::g()

D. B::f() D1::f() B::f() B::g() D1::g() D2::g()

E. B::f() B::f() D2::f() B::g() B::g() D2::g()

12. Ce se va afișa pe ecran în urma execuției următorului program:

```
#include <iostream.h>
class B {
public:
    virtual void f() { cout<<"B::f() ";}
};
class D1: public B{
public:
    void f() { cout<<"D1::f() ";}
};
class D2: public B{
public:
    void f() { cout<<"D2::f() ";}
};
void main(){
    int i;
    B b[] = { B(), D1(), D2()};
    for (i=2;i>=0;i--) {b[i].f();}
}
```


A. B::f() B::f() B::f()

B. D2::f() D1::f() B::f()

C. B::f() D1::f() D2::f()

D. B::f() D1::f() D2::f()

13. Ce se va afișa pe ecran în urma execuției următorului program:

```
#include <iostream.h>
class B {
protected:
    int x;
public:
    B(int x=0){this->x=x;}
    virtual void f() {x=x+1;}
    void print(){cout<<x<<" ";}
};
class D1: public B{
public:
    void f() {x=x+5;}
};
class D2: public B{
public:
    void f() {x=x+2;}
};
void main(){
    int i;
    B *b[] = { new B(2), new D1(), new D2(), new B};
    for (i=3;i>0;i--) (*b[i]).f();
    for (i=0;i<=3;i++) b[i]->print();
}
```

A. 2 5 2 1

B. 2 0 0 0

C. 3 5 2 1

D. 2 1 1 1

E. 2 5 2 0

F. 2 5 2

G. Eroare de compilare

14. Ce se va afișa pe ecran în urma execuției următorului program:

```
#include <iostream.h>
class B {
    int x;
public:
    B(int x=0){this->x=x;}
    virtual void f() {x=x+1;}
    void print(){cout<<x<<" ";}
};
class D1: public B{
public:
    void f() {x=x+5;}
};
class D2: public B{
```

```

public:
    void f() {x=x+2;}
};
void main(){
    int i;
    B *b[] = { new B(2), new D1(), new D2(), new B};
    for (i=3;i>0;i--) (*b[i]).f();
    for (i=0;i<=3;i++) b[i]->print();
}

```

A. 2 5 2 1

B. 2 0 0 0

C. 3 5 2 1

D. 2 1 1 1

E. 2 5 2 0

F. 2 5 2

G. Eroare de compilare

15. Ce se va afișa pe ecran în urma execuției următorului program:

```

#include <iostream.h>
class B {
    int x;
    static int y;
public:
    B(int x=0){this->x=x;}
    void setX(int vx) {x=vx;}
    void setY(int vy) {y=vy;}
    void print(){cout<<x<<" "<<y<<" ";}
};
int B::y=0;
void main(){
    B b1=3;
    B b2(4);
    b1.print(); b2.print();
    b1.setX(5); b2.setY(2);
    b1.print(); b2.print();
}

```

A. 3 0 4 0 5 2 4 2

B. 3 0 4 0 5 0 4 2

C. 3 4 5 2 4 2

D. 3 3 4 4 5 2 4 2

E. 3 0 4 0 5 2 5 2

F. 3 0 4 0 5 0 2 0

16. Ce se va afișa pe ecran în urma execuției următorului program?

```

#include <iostream.h>
struct A{
    int x;
    operator double() {

```

```

    return 21.4;
}
};
void main(){
    A a;
    a.x = 11;
    cout << (0?3:a);
}

```

- A. 21
- B. 21.4
- C. 3
- D. 11
- E. programul nu se compilează

17. Ce se va afișa pe ecran în urma execuției următorului program?

```

#include <iostream.h>
struct A {
    A(int d) : x(d) {}
    int x;
};
void main(){
    double x = 3.14;
    A f( int(x) );
    cout << f.x << endl;
}

```

- A. nimic (eroare de compilare)
- B. 0
- C. 3
- D. 3.14
- E. depinde de implementarea compilatorului

18. Ce se va afișa pe ecran în urma execuției următorului program?

```

#include <iostream.h>

int f(int x, int y = x) {
    return x+y+1;
}

void main(){
    cout << f(2);
}

```

- A. nimic (eroare de compilare)
- B. 1
- C. 3
- D. 5
- E. depinde de implementarea compilatorului

19. Ce se va afișa pe ecran în urma execuției următorului program?

```
#include <iostream.h>
struct A {
    virtual int f(int x = 5){
        return x * 2;
    }
};

struct B : public A{
    int f(int x = 10) {
        return x * 3;
    }
};

void main(){
    A* a = new B;
    cout << a->f();
}
```

- A. 15
- B. 10
- C. 20
- D. 3
- E. nimic (eroare de compilare)

20. Fie secvența de program:

```
class A {
    private:
        int x,y;
    public:
        A(){x=0;y=0; }
        A(int xi,int yi){
            x=xi;y=yi;
        }
        void afis(){
            cout <<x " " << y;
        }
};

void main(){
    A A1,A2(10,20);
    A1.afis();
    A2.afis();
}
```

Care din următoarele afirmații sunt adevărate:

- A. Metoda afis() este incorect definită
- B. Constructorul este incorect definit
- C. Declarație incorectă pentru obiectul A1
- D. Programul afisează valorile (10 20 0 0)

E. Programul afisează valorile (0 0 10 20)

21. Fie secvența de program:

```
class C{
    public:
        C()n++;
        static int index(){return n;}
    private:
        static int n;
};
int C :: n = 0;
class A{
    private:
        C c;
        int a;
};
class B{
    public:
        B(int i = 0) :b(i)
        {cout <<C :: index() <<" ";}
    private:
        A a;
        C c;
        int b;
};
void main(){
    A a1;
    cout <<C :: index() <<' ';
    B b1[3];
}
```

Rezultatul execuției programului este:

A. 1 2 3 4

B. 1 3 4 5

C. 1 7 3 4

D. 1 3 5 7

E. 0 1 2 3

22. Se considera secvența de program:

```
class C {
    private:
        int x,y;
    public:
        C(int xi,int yi){ x=xi;y=yi; }
        C(const C &a){
            x=xa.x;y=a.y;
```

```
    }  
};
```

În care din următoarele situații se realizează copierea unui obiect în altul:

A. C c1(4,5)

B. C c2(0.0, 0,0)

C. C c3=c1

D. C c4(1)

E. C c5(c1)

23. Se considera secvența de program:

```
class Punct {  
    private:  
        double x,y;  
    public:  
        Punct(double xi,double yi){  
            x=xi;y=yi;  
        }  
};
```

În care din următoarele situații se realizează o copiere a unui obiect:

A. Punct P1(10,20)

B. Punct P2(P1(3))

C. Punct P3

D. Punct P4=P1

E. Punct P5(P1)

24. Se dă programul :

```
class P {  
    double x,y;  
    public:  
        P(double x1,double y1){  
            x=x1;y=y1;  
        }  
        void afis(){  
            cout <<x " " << y;  
        }  
        P operator++ (){  
            ++x;++y;  
            return *this;  
        }  
};  
void main(){  
    P p1(1,2),p2;  
    p2=++p1;  
    p2.afis();  
}
```

Care din afirmațiile următoare sunt corecte:

A. Obiectul p2 nu exista

B. Asignarea p2=++p1 este incorectă

C. În program se realizează supraîncărcarea operatorului prefix de incrementare

D. Programul va afișa valorile: 2 3

E. Programul afiseaza valorile lui p1

```
25. class A {
    int x,y;
public:
    A(int xi,int yi){
        x=xi;y=yi;
    }
    void afis(){
        cout <<x " " << y;
    }
    A operator+ (A o2)
    friend A operator++ (A &a)
};
A A::operator+ (A o2){
    A temp;
    temp.x=x+o2.x;
    temp.y=y+o2.y;
    return temp;
}
A A::operator++ (A &a){
    ++a.x;++a.y;
    return a;
}
void main(){
    A a1(20,30),a2(2,2),a3;
    a3=++a1+a2;
    a3.afis();
}
```

Care din afirmațiile următoare sunt corecte:

A. Utilizarea cuvântului predefinit " friend " nu este permisă în acest context

B. Funcția operatorului ++ nu este definită corect

C. Programul va afișa valorile: 23 33

D. Atribuirea a3=++a1+a2 este incorectă

E. programul va afișa valorile: 22 32

26. Supraîncărcarea unor operatori se poate realiza prin funcții operator sau funcții friend.Diferența constă în:

A. Precedența operatorilor

B. Asociativitatea operatorilor

C. Obiectul returnat

D. Lista de parametri

E. Numărul parametrilor din lista funcției

27. Se considera secvența:

```
class A {  
    int a[3];  
public:  
    A(int xi, int yi, int zi){  
        a[0]=xi;a[1]=yi;a[2]=zi;  
    }  
    int &operator[](int i){  
        return a[i];  
    }  
};
```

```
void main() {  
    A o(1, 2, 3);  
    cout<<o[0];  
    o[1]=10;  
    cout<<o[1];  
}
```

} Ce se poate afirma despre operator[]()?

A. Supraîncărcă operatorul()

B. Supraîncărcă un operator unar

C. Este o metoda oarecare a clasei care nu produce supraîncărcarea unor operatori

D. Este un constructor

E. Supraîncărcă operatorul[]

28. Se consideră secvența:

```
class B{  
    int a;  
protected:  
    int b;  
public:  
    int c;  
    void set_a(int x){a = x;}  
    void set_b(int y){b = y;}  
    void set_c(int z){c = z;}  
};  
class D : public B{    int d;  
public:  
    void set_b(int y) {b = y;}  
    void set_c(int z) {c = z;}  
};  
void main(){
```



```

D o;
    o.a = 1;        //(1)
    o.B::set_a(2); //(2)
    o.b = 3;        //(3)
    o.B::set_b(4); //(4)
    o.c = 5;        //(5)
}

```

Care din instrucțiunile (1)-(5) accesează corect membrii claselor:

- A. Toate
- B. (1), (2) și (5)
- C. (1), (3) și(4)
- D. (1), (2), (3), (4)
- E. (2), (4) și (5)**

29. În programul următor, care din instrucțiunile (1)-(5) accesează corect membrii claselor: class B{

```

    int a;
protected:
    int b;
public:
    int c;
    void set_a(int x){a = x;}
    void set_b(int y){b = y;}
    void set_c(int z){c = z;}
};
class D : protected B{
    int d;
public:
    void set_b(int y) {b = y;}
    void set_c(int z) {c = z;}
};
void main(){
D o;
    o.a = 1;        //(1)
    o.B::set_a(2); //(2)
    o.b = 3;        //(3)
    o.B::set_b(4); //(4)
    o.c = 5;        //(5)
}

```

- A. Toate
- B. Nici una
- C. Numai (5)**
- D. (2), (4) și (5)

30. În programul următor, care din instrucțiunile (1)-(6) accesează corect membrii claselor:

```
class B{
    int a;
protected:
    int b;
public:
    int c;
    void set_a(int x){a = x;}
    void set_b(int y){b = y;}
    void set_c(int z){c = z;}
};
class D : private B{
    int d;
public:
    void set_b(int y) {b = y;}
    void set_c(int z) {c = z;}
};
void main(){
D o;
    o.a = 1;        //(1)
    o.B::set_a(2);  //(2)
    o.b = 3;        //(3)
    o.B::set_b(4); //(4)
    o.c = 5;        //(5)
    o.set_c = 6;    //(6)
}
```

- A. (1) și (4)
- B. (2), (3), (4)
- C. Toate
- D. (5) și (6)**

4 Programare orientata pe obiecte(Java)

1. Fie următoarea declarație Java:

```
public private int h;
```

Care afirmații sunt adevărate:

- A. Variabila h va fi accesată în mod public, deoarece se ia în considerare primul modificador de acces;
- B. Variabila h va fi accesată în mod private, deoarece se ia în considerare ultimul modificador de acces;
- C. Va fi eroare la compilare deoarece o variabilă nu poate fi în același timp accesată public și private;**
- D. Nici una din variantele de mai sus;

2. Fie următorul cod Java:

```
int x=0;
if (double.isInfinite(2/x))
System.out.println("Infinit");
else
System.out.println("2/0");
```

Ce puteți spune despre acest cod, dacă este integrat în cadrul unui program Java?

- A. Va produce eroare la compilare din cauza împărțirii la 0;
- B. Va produce eroare la execuție din cauza împărțirii la 0 (se aruncă o excepție: "ArithmeticExpetion");**
- C. Codul este corect și va afișa Infinit;
- D. Codul este corect și va afișa NaN;

3. Fie următorul program Java:

```
public class Program{
static void f(int k){
switch(k){
default: System.out.print("i "); break;
case1: System.out.print("1 "); break;
case 2:case3: System.out.print("23 "); break;
case 4:case5: System.out.print("45 ");
}
}
public static void main(String []args){
for(int i=0;i<6;i++){
f(i);
}
}
```

Care afirmații sunt adevărate?

- A. Programul produce eroare la compilare;
- B. Programul se compilează și la execuție afișează
i 1 23 23 45 45 ;**
- C. Programul se compilează și la execuție afișează
i 1 23 45 ;

D. Programul se compilează și la execuție afișează

```
i 1 23 23 45 45 i;
```

4. Fie următorul cod Java:

```
byte b=-7 >>> 1;  
System.out.println(b);
```

Ce se poate spune despre acest cod, dacă este integrat într-un program Java?

A. Va produce eroare la compilare;

B. Va produce eroare la execuție;

C. Programul se compilează și la execuție afișează -3;

D. Programul se compilează și la execuție afișează -4;

5. Ce puteți afirma despre următorul program Java?

```
public class Static1{  
    public static void main(String []args){  
        Static2 a=new Static2();  
        Static2 b=new Static2();  
        System.out.print("a.x= "+ a.x);  
        a.x=100; b.x=200;  
        System.out.print("a.x= "+a.x);  
    }  
}  
class Static2{  
    static int x=0;  
    Static2()  
    {  
        x++;  
    };  
}
```

A. Afiseaza:

```
a.x=2 a.x=200;
```

B. Afiseaza:

```
a.x=0 a.x=100;
```

C. Afiseaza:

```
a.x=1 a.x=100;
```

D. Programul nu este corect deoarece asignarea lui x, conform obiectului b, este ilegală în Java;

6. Ce se va afișa la execuția următorului program Java?

```
interface I1{  
    float x=2.3f;  
}  
public class Test implements I1{  
    public static void main(String [] args){  
        System.out.print(x+" ");  
        x=6.7f;  
        System.out.print(x);  
    }  
}
```

A. Va apărea eroare la compilare deoarece valoarea variabilei x nu se mai poate modifica;

B. La execuție se va afișa:

2.3f 6.7f;

C. La execuție se va afișa:

2.3f 2.3f;

D. La execuție se va afișa:

2.3 6.7;

7. Ce puteți spune despre următorul program Java?

```
class C1{
int x=1;
void f(int x){
this.x=x;
}
int getX_C1(){
Return x;
}
}
Class C2 extends C1{
float x=5.0f;
int f(int x){
super.f((int)x);
}
float getX_C2(){
return x;
}
}
public class Subiect9{
public static void main(String []args){
C2 obiect = new C2();
obiect.f(4);
System.out.print(obiect.getX_C2() + " ");
System.out.println(obiect.getX_C1());
}
}
```

A. Programul este corect și va afișa la execuție 5 4;

B. Programul este corect și va afișa la execuție 4.0 4;

C. Va apărea eroare la compilare deoarece în clasa C2 s-a suprascris greșit atributul x din clasa C1;

D. Va apărea eroare la compilare deoarece metoda suprascrisă f() din clasa C2 întoarce un tip diferit de void;

8. Ce puteți spune despre următorul program Java?

```
public class Test{
public static void main(String []args){
C1 obiect =new C1();
obiect.f(4,3);
}
}
class C1{
public void f(int xx, final int yy){
```

```

int a=xx+yy;
final int b=xx-yy;
class C2{
public void g(){
System.out.print("a= "+a);
System.out.print(", b= "+b);
}
}
C2 obiect2 = new C2();
obiect2.g();
}
}

```

- A. Programul este corect și va afișa la execuție a=4, b=3;
 - B. Va apărea eroare la compilare, deoarece clasa C2 nu poate fi definită în metoda f() din clasa C1;
 - C. Va apărea eroare la compilare deoarece în metoda g() nu putem accesa variabila locală a din metoda f();**
 - D. Va apărea eroare la compilare deoarece nu se creează în clasa Test un obiect de tip C1.C2;
9. Un fir de execuție poate intra în starea "blocat" (blocked) astfel?
- A. Prin apelul metodei *sleep()*;**
 - B. Automat de către sistemul de operare;
 - C. Prin apelul metodei *block()*;
 - D. Prin apelul metodei *wait()*;**
10. Care dintre următoarele propoziții referitoare la metodele unui applet sunt adevărate?
- A. Trebuie să redefinim măcar o metodă altfel obținem eroare la compilare;
 - B. Sunt apelate automat de navigatorul Web;**
 - C. Pot fi apelate direct de către utilizator;**
 - D. Nu se pot declara noi metode;
11. Care dintre gestionarii de poziționare (Layout Managers) de mai jos pot fi utilizați pentru applet-uri?
- A. GridBagLayout;**
 - B. TableLayout;
 - C. DefaultLayout;
 - D. FlowLayout;**
12. Ce metode trebuie definite pentru a putea desena pe suprafața applet-ului?
- A. Nu trebuie definită nici o metodă;
 - B. *update()*;
 - C. *paint()*;**
 - D. *repaint()*;
13. Care sunt deosebirile dintre Swing și AWT?
- A. Componentele Swing sunt scrise în totalitate în Java, pe când în AWT, componentele sunt scrise folosind cod nativ;**

- B. Componentele AWT au vizualizarea dependentă de sistemul de operare, iar în Swing componentele pot avea o aceeași vizualizare, indiferent de sistemul de operare;**
- C. În sistemele de operare Unix componentele AWT nu sunt vizibile, pe când cele din Swing da;
- D. Fiecare componentă AWT are o componentă corespondentă în Swing;
14. Care din fragmentele de cod Java de mai jos daugă un buton unui container de baza JFrame?
- A. `JFrame f= new JFrame();
JPanel p = (JPanel)f.getContentPane();
p.add(new JButton("Buton")) ;`**
- B. `JFrame f= new JFrame();
f.getContentPane().add(new JButton("Buton"));`**
- C. `JFrame f= new JFrame();
f.add(new JButton("Buton"));`
- D. `JFrame f=new JFrame();
JButton b=new JButton("Buton");
f.add(b);`
15. Fie următoarea secvență de cod:
- ```
JPanel p =new JPanel();
JLabel e = new JLabel("Eticheta:");
e.setDisplayedMnemonic('E');
p.add(et, BorderLayout.EAST);
JTextField t = new JTextField(7);
e.setLabelFor(t);
p.add(t, BorderLayout.WEST);
```
- Ce se va întâmpla la apăsarea combinației de taste: Alt + E?
- A. Eticheta e va deține focusul;
- B. Câmpul t va deține focusul;**
- C. Nici o componentă nu va deține focusul;
- D. Panoul p va deține focusul;
16. Care dintre următoarele afirmații referitoare la componenta grafică *JTable* sunt adevărate?
- A. Întotdeauna o componentă *JTable* are asociată un model de date, chiar dacă nu specificăm explicit acest lucru;**
- B. Prin modificările efectuate asupra datelor unui model, se va actualiza automat și vizualizarea tabelului;**
- C. Pentru a crea o tabelă folosind modelul *AbstractTableModel*, este îndeajuns să suprascriem metodele *getRowCount()* și *getColumnCount()*;
- D. Modelul de date *AbstractTableModel* ține datele întotdeauna într-un Vector având elemente Vector;
17. *JDesktopPane* este un exemplu de:
- A. Container pentru *JInternalFrame*;**
- B. Subclasă pentru *JLayeredPane*;**
- C. Subclasă pentru *JInternalFrame*;
- D. Componentă atomică simplă;

18. Fie următorul program Java:

```
public class Afisare {
 public static void main (String[] args) {
 for (int i = 0; i < args.length; i++)
 System.out.println(args[i]);
 }
}
```

Un apel de genul `java Afisare "Hello Java"`; va produce următorul rezultat:

**A. Hello Java;**

B. Hello  
Java;

C. Programul este incorect, deoarece nu sunt prezente argumentele;

D. NaN;

19. Fie următorul program Java:

```
public class Program{
 static void f(int k){
 switch(k){
 default: System.out.print("i "); break;
 case1: System.out.print("1 "); break;
 case 2:case3: System.out.print("21 "); break;
 case 4:case5: System.out.print("26 ");
 }
 }
 public static void main(String []args){
 for(int i=0;i<6;i++)
 f(i);
 }
}
```

Care afirmații sunt false?

**A. Eroare la compilare;**

B. Programul se compilează și la execuție afișează  
i 1 21 21 26 26 ;

**C. Programul se compilează și la execuție afișează**

**i 1 21 26 ;**

**D. Programul se compilează și la execuție afișează**

**i 1 21 21 26 26 i;**

20. Se utilizează `protected` în Java pentru date și metode la care nu este necesar:

A. Să facem o inițializare;

B. Să le accesăm în subclase;

C. Ca utilizatorul să aibă acces;

**D. Accesul direct atunci când clasa este utilizată dar care prezintă interes atunci când cineva creează o subclasă a acesteia ca parte a unui pachet diferit, în vederea extinderii ei;**

21. O subclasă a unei clase abstracte poate fi instanțiată numai dacă:

A. Se folosește cuvântul cheie `abstract`;



- B. Suprascrie fiecare metodă declarată abstractă în superclasa sa, și furnizează implementări pentru toate acestea;**
- C. Se folosește moștenirea multiplă;
- D. O subclasă abstractă nu poate fi instanțiată;
22. Prin modalitatea sa de tratare a excepțiilor, Java are următoarele avantaje față de mecanismul tradițional de tratare a erorilor:
- A. Există o metodă care se ocupă cu acest lucru;
- B. Separarea codului pentru tratarea unei erori de codul în care ea poate să apară;**
- C. Propagarea unei erori până la un analizor de excepții corespunzător;**
- D. Gruparea erorilor după tipul lor;**
23. Metodele care sunt apelate uzual pentru un obiect de tip *excepție* sunt definite în clasa *Throwable* și sunt:
- A. Declarate cu modificatorul de acces *private*;
- B. dinamice;
- C. publice;**
- D. excepții;
24. Un fir de execuție poate intra în starea de *ready* astfel:
- A. Prin apelul metodei *sleep()*;
- B. Automat de către sistemul de operare;
- C. Prin apelul metodei *join()*;
- D. Niciodată;
25. Când browser-ul întâlnește tag-ul `<APPLET >`, rezervă o zonă pentru afișare cu dimensiunile specificate de paramterii *WIDTH*, *HEIGHT* și:
- A. Se instalează un manager de securitate, adică un obiect de tip *SecurityManager* care va monitoriza activitatea metodelor appletului, aruncând excepții de tip *SecurityException*;
- B. Încarcă codul compilat al applet-ului cu numele specificat de parametrul *CODE*;
- C. Crează o instanță a clasei *Applet* după care apelează metodele *init()* și *start()*;**
- D. Se deschid mai multe procese pe mașina client;
26. Care dintre următoarele coduri nu reprezintă arhivarea fișierelor unui applet?
- A. jar cvf arhiva.jar ClasaPrincipala.java imagine.jpg;**
- B. jar cvf arhiva.jar \*.class \*.jpg \*.au;
- C. jar cfv arhiva.jar \*.class \*.jpg \*.au;**
- D. jar cvf arhiva.jar ClasaPrincipala.class AltaClasa.class imagine.jpg sunet.au
27. În care din exemplele de mai jos se folosește corect variabila *iLocation*?
- A. `tabbedPanel.insertTab( "Inserted Page", new ImageIcon( "image.gif" ), pagePanel,"My tooltip text",iLocation );`**
- B. `JFrame f= new JFrame();  
f.getContentPane().add(new JButton("Buton", iLocation));`
- C. `tabbedPanel.removeTabAt( iLocation );`**

```
D. JFrame f=new JFrame();
 JButton b=new JButton("Buton");
 f.add(b, iLocation);
```

28. Ce rezultă din următorul fragment de cod Java?

```
int x=1;
String []names={"Fred","Jim","Sheila"};
names[--x]+=".";
for(int i=0;i<names.length;i++)
 System.out.println(names[i]);
```

- A. Output-ul include Sheila.;
- B. Output-ul include Fred.;**
- C. Output-ul include Jim.;
- D. Nimic din cele de mai sus;

29. Applet-urile se diferențiază de aplicațiile Java standard prin:

- A. Restricțiile impuse de necesitatea asigurării unui anumit nivel de securitate și faptul că nu au o metodă *main()*;**
- B. Faptul că trebuie să suprascrise toate metodele: *init()*, *start()*, *stop()*, *pause()* și *destroy()*;
- C. Faptul că dețin un instrument cu ajutorul căruia se crează relații de moștenire;
- D. Faptul că dețin un instrument care desemnează meniurile din cadrul unei forme;

30. Declararea constructorilor trebuie să țină cont de:

- A. relația de moștenire dintre clase;
- B. numele constructorului, care trebuie să fie identic cu numele clasei;**
- C. comportamentul obiectelor pe care le instanțiază;
- D. o metoda prin care poate fi accesat de toate tipurile din Java sau de tipuri moștenite din tipul care conține membrul în discuție;