A. Deduceți timpii mediu si defavorabil pentru subalgoritmul **transformare**. Justificați rezultatul.

```
Subalgoritmul g(x, n, y, i) este
|       {pre: x șir de întregi cu n elemente; i: Intreg}
|       {post: y șir de întregi}
|       dacă i ≤ n atunci
|       |       y[i] ← x[i]
|       |       g(x, n - 1, y, i + 1)
|       Sfdacă
Sfg

Subalgoritmul transformare(x, n, y, m, z, k,) este
|       {pre: x șir de întregi cu n elemente; y șir de întregi cu m elemente }
|       {post: z șir de întregi cu k elemente}
|       dacă n = 0 atunci
|               g(y, m, z, 1)
|               k ← m
|       altfel
|               y[m +1] ← x[n]
|               transformare(x, n - 1, y, m + 1, z, p)
|               k ← p
|       Sfdacă
Sftransformare
```

$$T_g(n,i) = \begin{cases} T_g(n-1, i+1) + 1, & i \le n \\ 0, & i > n \end{cases}$$

$$T_g(n,i) = T_g(n-1, i+1) + 1 = T_g(n-2, i+2) + 2$$

$$T_g(n-1, i+1) = T_g(n-2, i+2) + 1$$

$$\Rightarrow T_g(n,i) = T_g(n-k, i+k) + k$$

$$n - k = i + k$$

$$\Rightarrow k = \frac{n-i}{2} \quad \Rightarrow T_g(n,i) = T(\underbrace{n-k-1, n-k+1}_{i > n}) + \frac{n-i}{2}$$

$$= \frac{n-i}{2} \in \Theta(n)$$

$$T_f(n,m) = \begin{cases} T_g(m,1), & n = 0 \\ T_f(n-1, m+1) + 1, & \text{altfel} \end{cases}$$

$$T_f(n,m) = T_f(n-1, m+1) + 1$$

$$T_f(n-1, m+1) = T_f(n-2, m+2) + 1$$

$$\Rightarrow T_f(n,m) = T_f(n-k, m+k) + k$$

$$n = k$$

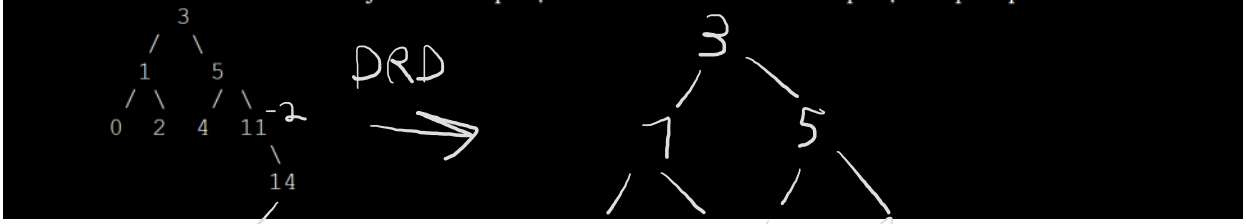
$$\Rightarrow T_f(n,m) = T_f(0, m+k) + n$$

$= T_g(m+k, 1) + n$

$= \frac{m+k-1}{2} + n \in \Theta(m+n)$

$\frac{k-1}{2}$ constantă

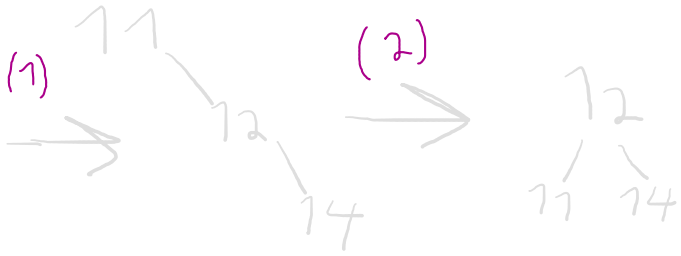
B. Cum arată arborele AVL de mai jos în urma operației de inserare a cheii 12?. Ce operație se aplică pentru reechilibrare?



12

se face o rotire la dreapta a lui 14 (1)

— " — stânga a lui 11 (2)



C. Alegeți afirmația corectă. Justificati

a) orice arbore binar este fie complet, fie plin
b) orice arbore binar plin este și complet
c) niciun arbore binar nu poate fi și complet și plin
d) orice arbore binar complet este plin

1. Arbore binar plin (*full binary tree*)

- Fiecare **nod intern** are exact **doi copii**.
- **Nu există noduri** cu un singur copil.
- Frunzele (noduri fără copii) pot fi pe **niveluri diferite**.

Exemplu:

mathematica Copy Edit

```
    A
   / \
  B   C
 / \ / \
D  E F  G
```

2. Arbore binar complet (*complete binary tree*)

- Toate nivelurile sunt **complet umplute, cu excepția ultimului**.
- Pe **ultimul nivel**, nodurile sunt adăugate **de la stânga la dreapta**, fără „găuri".

Exemplu:

mathematica Copy Edit

```
  A
 / \
B   C
/ \ /
D E F
```

→ nu trebuie să fie găuri

✅ Toate nivelurile sunt pline, în afară de ultimul.

✅ Pe ultimul nivel, nodurile sunt **cât mai la stânga** → este complet.

1. Arbore binar plin (*full binary tree*)

- Fiecare **nod intern** are exact **doi copii**.
- **Nu există noduri** cu un singur copil.
- Frunzele (noduri fără copii) pot fi pe **niveluri diferite**.

Exemplu:

mathematica Copy Edit

```
      A
     / \
    B   C
   / \ / \
  D  E F  G
```

2. Arbore binar complet (*complete binary tree*)

- Toate nivelurile sunt **complet umplute, cu excepția ultimului**.
- Pe **ultimul nivel**, nodurile sunt adăugate **de la stânga la dreapta**, fără „găuri".

Exemplu:

mathematica Copy Edit

```
    A
   / \
  B   C
 / \ /
D  E F
```

→ nu trebuie să fie găuri

- ✅ Toate nivelurile sunt pline, în afară de ultimul.
- ✅ Pe ultimul nivel, nodurile sunt **cât mai la stânga** → este **complet**.

C1. b) Într-un arbore binar plin, fiecare nivel fără nod complet, înseamnă că fiecare nod intern are exact 2 fii

pe ultimul nivel

C. Presupunem o Colectie implelementată folosind o listă înlănțuită. Care din operațiile de mai jos au complexitatea defavorabilă θ(1)? Justificati

a) adăugare b) ștergere c) numărAparitii

✅ Justificare (într-o propoziție):

Adăugarea unui element la începutul unei liste înlănțuite are complexitate defavorabilă $\theta(1)$, deoarece presupune doar alocarea unui nou nod și legarea lui de restul listei, fără a parcurge elementele.