ντήστε στις παρακάτω ερωτήσεις. Δεν χρειάζεται να δικαιολογήσετε την απάντηση σας. Λάθος
ώστε τις σωστές απαντήσεις (ενδέχεται να υπάρχει παραπάνω από μία σωστή απάντηση ή
ς και καμία).

1. Ποιοι από τους παρακάτω ισχυρισμούς είναι σωστοί;
   a. $n^4 + 3n^3\log n - 10n^2 = O(n^3 (\log n)^2)$
   b. $10n^2(\log n)^2 + 3n^3\log n + 5n^4 + 10n^2 = O(n^3(\log n)^2)$
   c. $8n^5 + 8n^4 + 30n^3 + 100n^2 = O(n^5)$
   d. $\text{sqrt}(n) = O(n/\log n)$, όπου sqrt(n) είναι η τετραγωνική ρίζα του n
   e. $\log_M n = O(1)$, όπου $M = O(1)$
   f. $n\log_3(n^7) = O(n\log n)$
   g. $\log_3(n^n) = O(n^{\log n})$
   h. $(2n + \log n)^2 = O(n(\log n)^2)$

2. Οι μέθοδοι ταξινόμησης Insertionsort και Mergesort έχουν
   a. τις ίδιες απαιτήσεις μνήμης
   b. την ίδια πολυπλοκότητα χειρότερης περίπτωσης
   c. την ίδια πολυπλοκότητα μέσης περίπτωσης

3. Σε ένα δέντρο κόκκινου-μαύρου (με cbit το πεδίο που δείχνει το χρώμα ενός συνδέσμου), ένας 4-κόμβος που ξεκινάει από την αναφορά h, ικανοποιεί
   a. `h.cbit = R, h.l.cbit = R, h.r.cbit = R`
   b. `h.cbit = B, h.l.cbit = R, h.r.cbit = R`
   c. `h.cbit = R, h.l.cbit = B, h.r.cbit = B`

4. Μπορούμε να κατασκευάσουμε ένα δέντρο 2-3-4 με ύψος 1 με τον αλγόριθμο ανοδικής εισαγωγής, που να περιέχει 14 κλειδιά.
   a. Σωστό
   b. Λάθος

5. Έστω ένα δέντρο 2-3-4 και ένα AVL δέντρο με τον ίδιο αριθμό αντικειμένων. Έστω h το ύψος του δέντρου 2-3-4. Το ύψος του AVL δέντρου είναι πάντα
   a. τουλάχιστον h
   b. τουλάχιστον 2h
   c. μικρότερο ή ίσο από 2h

6. Ένας δυαδικός σωρός που έχει 4 επίπεδα κόμβων, είναι εφικτό να έχει
   a. 15 αντικείμενα
   b. 7 αντικείμενα
   c. 11 αντικείμενα

(β) [5 μονάδες] Εκτελούμε την εξής μέθοδο, η οποία δέχεται ως είσοδο έναν μη αρνητικό ακέραιο αριθμό:

```
void foo(int n) {
if (n==1) System.out.println("Recursion or not?");
if (n>1) {
        System.out.println("Recursion or not?");
        foo(n/2);
    }
}
```

Πόσες φορές (ως συνάρτηση του n, με συμβολισμό μεγάλου O) θα τυπώσει η μέθοδος την πρόταση *"Recursion or not?"*;

Για να απαντήσετε την ερώτηση, πρέπει να γράψετε πρώτα μια αναδρομική σχέση που περιγράφει τον αριθμό των γραμμών που τυπώνονται. Κατόπιν, εξηγήστε ποια είναι η λύση της αναδρομικής εξίσωσης. Για ευκολία, μπορείτε να υποθέσετε ότι η είσοδος n, είναι πάντα δύναμη του 2.

ΘΕΜΑ 3 (24 μονάδες)

Εκτιμώμενος χρόνος επίλυσης: 25-30 λεπτά

(α) [12 μονάδες] Ξεκινώντας από έναν άδειο μεγιστοστρεφή δυαδικό σωρό, να κάνετε εισαγωγή των εξής στοιχείων (με τη σειρά που δίνονται): 8, 20, 15, 13, 9, 25, 22, 28, 26. Να σχεδιάσετε το δέντρο που προκύπτει, τη στιγμή που έχει ολοκληρωθεί η εκάστοτε εισαγωγή (συνολικά πρέπει να σχεδιάσετε 9 δέντρα). Αν σας διευκολύνει στην παρουσίαση, μπορείτε να σχεδιάσετε προαιρετικά και τα ενδιάμεσα δέντρα που προκύπτουν μετά από κάθε ανάδυση, κατά τη διάρκεια των εισαγωγών.

(β) [5 μονάδες] Εκτελέστε μία αφαίρεση μεγίστου (`getmax`) στο σωρό που έχει προκύψει όταν ολοκληρώνονται όλες οι εισαγωγές στο ερώτημα (α). Δείξτε όλες τις αλλαγές που γίνονται (μετακινήσεις στοιχείων) μέχρι να ολοκληρωθεί η κλήση της `getmax`.

(γ) [7 μονάδες] Έστω ένας μεγιστοστρεφής δυαδικός σωρός με ακέραια κλειδιά και με τις εξ ιδιότητες: ο σωρός έχει 7 κόμβους, και περιέχει 3 επίπεδα, το επίπεδο 0 που περιέχει τη ρίζα, επίπεδο 1 και το επίπεδο 2. Η μεταδιατεταγμένη διάσχιση στο δέντρο επισκέπτεται τελευταίο κλειδί 38. Στην ενδοδιατεταγμένη διάσχιση, το πρώτο κλειδί που επισκεπτόμαστε είναι το 32 κ προτελευταίο είναι το 31. Τέλος, η προδιατεταγμένη διάσχιση επισκέπτεται τέταρτο το κλειδ Σχεδιάστε ένα σωρό που ικανοποιεί όλες αυτές τις ιδιότητες (υπάρχουν αρκετές πιθανές λ εσείς απλά επιλέξτε μια τιμή για κάθε κόμβο του σωρού, έτσι ώστε να ισχύουν οι ιδιό Αιτιολογήστε συνοπτικά (3-4 γραμμές αρκούν) την απάντησή σας.

ΘΕΜΑ 2 (15 μονάδες)

Εκτιμώμενος χρόνος επίλυσης: 20 λεπτά

(α) [10 μονάδες] Δίνεται ο ακόλουθος ορισμός των κόμβων μίας απλά συνδεδεμένης λίστας.

```
class Node{
    int key;
    Node next;
    Node(int x){ key = x; next = null; } }
```

Να γράψετε σε Java τη μέθοδο `Node delete(Node h, int x)` η οποία κάνει αναζήτηση και μετέπειτα διαγραφή ενός κόμβου με κλειδί x στην ταξινομημένη λίστα (σε αύξουσα σειρά) με κεφαλή h. Η μέθοδος θα πρέπει να δουλεύει ως εξής: αν το κλειδί x υπάρχει στη λίστα, η μέθοδος θα πρέπει να διαγράψει τον πρώτο κόμβο που θα βρει ότι έχει το κλειδί x, και θα επιστρέψει την κεφαλή της λίστας που απέμεινε μετά την διαγραφή (μπορεί να υπάρχουν κι άλλοι κόμβοι με το ίδιο κλειδί αλλά δεν μας απασχολεί αυτό). Διαφορετικά η μέθοδος επιστρέφει null. Η λίστα δεν έχει κόμβους φρουρούς και τερματίζεται με null.