ΠΑΡΑΡΤΗΜΑ: QUIZ – ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

ΕΙΣΑΓΩΓΗ

Q1. **Οι Πρωταρχικοί τύποι** (primitive types) στη Java

1. Είναι όλοι οι ακέραιοι και όλοι οι πραγματικοί τύποι
2. Είναι όλοι οι αριθμητικοί τύποι, ο τύπος **char** και ο τύπος **boolean**
3. Είναι ο τύπος **byte** και ο τύπος **char**
4. Είναι ο τύπος **int** και ο τύπος **char**

Q2. Ποιοι θεωρούνται **τύποι αναφοράς** (reference types) στη γλώσσα Java;

1. οι **τύποι κλάσης** (class types)
2. οι **τύποι κλάσης** και οι **τύποι πίνακα** (array types)
3. οι **τύποι κλάσης** και οι **τύποι διασύνδεσης** (interface types)
4. οι **τύποι κλάσης**, οι **τύποι πίνακα** και οι **τύποι διασύνδεσης**

Q3. Στο παρακάτω τμήμα προγράμματος Java:

```
int x = 109;
char ch = 'Z';
if (x > ch) { . . . }
```

1. Θα προκληθεί σφάλμα μεταγλώττισης λόγω ασυμβατότητας τύπων των μεταβλητών **x** και **ch**.
2. Η τιμή της μεταβλητής **x** μεταπίπτει σε χαρακτήρα και στη συνέχεια υπολογίζεται η boolean τιμή της σύγκρισης στο **if**
3. Η τιμή της μεταβλητής **ch** μεταπίπτει σε ακέραιο και στη συνέχεια υπολογίζεται η boolean τιμή της σύγκρισης στο **if**
4. Η boolean τιμή της σύγκρισης στο **if** έχει πάντοτε την τιμή **false**.

ΓΡΑΜΜΙΚΕΣ ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

Q4. Στη δομή δεδομένων πίνακας το μέγεθος του (μήκος)

1. Είναι σταθερό
2. Ορίζεται από το χρήστη με την κλήση της μεθόδου length()
3. Ορίζεται από το χρήστη με την ενημέρωση του πεδίου length
4. Ορίζεται αυτόματα από το σύστημα της JAVA στο χρόνο εκτέλεσης

Q5. Γραμμική δομή δεδομένων (*Linear data type*) ονομάζεται η δομή τα στοιχεία της οποίας είναι διατεταγμένα με τέτοιο τρόπο ώστε:

1. Να βρίσκονται σε διαδοχικές θέσεις μνήμης
2. Να συνδέονται μεταξύ τους με τη βοήθεια δεικτών
3. Το καθένα από αυτά να έχει ένα προηγούμενο και ένα επόμενο
4. Να υπάρχει ένα μόνο στοιχείο το οποίο έχει ένα επόμενο στοιχείο, ένα μόνο στοιχείο το οποίο έχει ένα προηγούμενο και κάθε άλλο στοιχείο έχει ένα προηγούμενο και ένα επόμενο

Q6. Η πρόταση **Vector v = new Vector(10, 5);** δημιουργεί:

1. Ένα διάνυσμα 10 θέσεων με των 5 bytes ανά θέση
2. Ένα διάνυσμα αρχικής χωρητικότητας 5 θέσεων και μέγιστης χωρητικότητας 10.
3. Ένα διάνυσμα αρχικής χωρητικότητας 10 θέσεων και μέγιστης χωρητικότητας 15.
4. Ένα διάνυσμα αρχικής χωρητικότητας 10 θέσεων, το οποίο κάθε φορά που απαιτείται αυξάνει τη χωρητικότητα του κατά 5 θέσεις.

Q7. Η διαφορά ανάμεσα στον τύπο δεδομένων **Πίνακας** (*Array*) και στον τύπο δεδομένων **Διάνυσμα** (*Vector*) είναι ότι:

1. Ο Πίνακας είναι Γραμμική Δομή Δεδομένων, ενώ το διάνυσμα δεν είναι.
2. Ο Πίνακας είναι δομή δεδομένων σταθερού μήκους ενώ το διάνυσμα δομή δεδομένων μεταβλητού μήκους.
3. Το διάνυσμα είναι Γραμμική Δομή Δεδομένων, ενώ ο Πίνακας δεν είναι.
4. Ο Πίνακας είναι δυναμική δομή δεδομένων ενώ το διάνυσμα στατική δομή δεδομένων.

Q8. Η διαφορά ανάμεσα στον τύπο δεδομένων **String** και στον τύπο **StringBuffer** είναι ότι:

1. Το **String** είναι αμετάβλητη Δομή Δεδομένων, ενώ το **StringBuffer** είναι ευμετάβλητη.
2. Το **String** είναι Γραμμική Δομή Δεδομένων, ενώ το **StringBuffer** δεν είναι.
3. Δεν έχουν διαφορά. Είναι η ίδια δομή δεδομένων που μπορεί να οριστεί στα πλαίσια δύο διαφορετικών κλάσεων.
4. Το **StringBuffer** είναι δομή δεδομένων σταθερού μήκους, ενώ το **String** δεν είναι.

Q9. Η κλάση **StringTokenizer** χρησιμοποιείται για:

1. Να αφαιρέσει τα κενά από μία συμβολοσειρά (string)
2. Να χωρίσει μία συμβολοσειρά, με βάση τους οριοθέτες που περιέχει, σε υποσειμβολοσειρές (sub-strings)
3. Να δημιουργήσει τα σύνολα των γραμμάτων και των ψηφίων μιας συμβολοσειράς
4. Να χωρίσει μία συμβολοσειρά σε υποσυμβολοσειρές ίσου μήκους

ΣΤΟΙΒΕΣ & ΟΥΡΕΣ

Q10. Το περιεχόμενο της στοίβας (stack) μετά την εκτέλεση της παρακάτω σειράς από πράξεις: **Π Α Ρ * Α * * Π Ο Λ * * Υ Ε * * * Υ * Κ Ο * Λ Ο *** είναι: (Ενα γράμμα ερμηνεύεται σαν πράξη εισαγωγής του γράμματος στη στοίβα, ενώ ο χαρακτήρας "*" ερμηνεύεται σαν πράξη εξαγωγής ενός γράμματος από τη στοίβα.)

1. **Ο Λ Ο Κ Υ Ε Υ Λ Ο Π Α Ρ Α Π**
2. **Π Ρ Π Ο Λ Υ Κ Λ**
3. **Π Κ Λ**
4. **Ο Λ Ο**

Q11. Το περιεχόμενο της ουράς (queue) μετά την εκτέλεση της παρακάτω σειράς από πράξεις: **Π Α Ρ * Α * * Π Ο Λ * * Υ Ε * * * Υ * Κ Ο * Λ Ο *** είναι: (Ενα γράμμα ερμηνεύεται σαν πράξη εισαγωγής του γράμματος στη στοίβα, ενώ ο χαρακτήρας "*" ερμηνεύεται σαν πράξη εξαγωγής ενός γράμματος από τη στοίβα.)

1. **Π Α Ρ Α Π Ο Λ Υ Ε Υ Κ Ο Λ Ο**
2. **Λ Κ Υ Λ Ο Π Ρ Π**
3. **Π Κ Λ**
4. **Ο Λ Ο**

Q12. Η υλοποίηση μίας ουράς με τη βοήθεια στοίβας είναι δυνατή:

1. ΟΧΙ γιατί η πρώτη είναι δομή LIFO και η δεύτερη δομή FIFO
2. ΝΑΙ αν χρησιμοποιηθούν δύο στοίβες
3. ΝΑΙ καθώς οι πράξεις εισαγωγής και διαγραφής των δύο δομών ταυτίζονται
4. ΟΧΙ καθώς οι πράξεις εισαγωγής και διαγραφής των δύο δομών διαφέρουν

Q13. Στην περίπτωση υλοποίησης της δομής δεδομένων **ουρά** (***queue***) με τη βοήθεια πίνακα η υπερχείλιση διαφέρει από την εικονική υπερχείλιση:

1. ΟΧΙ γιατί ο πίνακας έχει σταθερό μήκος
2. ΝΑΙ γιατί στην εικονική υπερχείλιση μπορεί να υπάρχει ελεύθερος χώρος στον πίνακα ενώ στην υπερχείλιση όχι.
3. ΟΧΙ γιατί και στις δύο περιπτώσεις το μήκος του πίνακα ταυτίζεται με το μήκος της ουράς που υλοποιεί.
4. ΝΑΙ καθώς όταν έχουμε εικονική υπερχείλιση η ουρά μετατρέπεται ουσιαστικά σε μία στοίβα.

Q14. Στην περίπτωση της υλοποίησης της ουράς με τη βοήθεια πίνακα, το πρόβλημα που λύνει η χρήση κυκλικής ουράς είναι ότι:

1. Καταργεί τα προβλήματα της εικονικής υπερχείλισης και της άδειας ουράς.
2. Καταργεί τα προβλήματα της υπερχείλισης και της άδειας ουράς.
3. Καταργεί τα προβλήματα της εικονικής υπερχείλισης και της υπερχείλισης.
4. Καταργεί το πρόβλημα της εικονικής υπερχείλισης.

ΔΥΝΑΜΙΚΕΣ ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

Q15. Η έννοια της **δυναμικής δομής δεδομένων *(Dynamic data type)*** ορίζεται ως:

1. Η δομή δεδομένων η οποία μπορεί να δημιουργηθεί και να μετατρέπεται στη διάρκεια της εκτέλεσης του προγράμματος αυξομειώνοντας το μέγεθός της ή/και το χώρο που καταλαμβάνει στη μνήμη
2. Η δομή δεδομένων στην οποία μπορούν να εισάγονται και να διαγράφονται στοιχεία κατά τη διάρκεια της εκτέλεσης του προγράμματος
3. Η δομή δεδομένων η οποία κατά τη διάρκεια της εκτέλεσης του προγράμματος αλλάζει συνεχώς θέση στη μνήμη.
4. Η δομή δεδομένων η οποία βρίσκεται στη μνήμη για όλη τη διάρκεια της εκτέλεσης του προγράμματος.

Q16. Το βασικό πλεονέκτημα μιας συνδεδεμένης λίστας σε σχέση με έναν πίνακα είναι ότι:

1. Η συνδεδεμένη λίστα καταλαμβάνει πολύ λιγότερο χώρο στη μνήμη από ότι ο πίνακας
2. Είναι πολύ πιο γρήγορη η πρόσβαση σε ένα οποιοδήποτε στοιχείο της λίστας σε σχέση με την πρόσβαση ενός στοιχείου του πίνακα.
3. Η συνδεδεμένη λίστα έχει μεταβλητό μέγεθος ενώ ο πίνακας όχι.
4. Είναι πιο εύκολη η ταξινόμηση της συνδεδεμένης λίστας.

Q17. Η υλοποίηση μιας ουράς με τη βοήθεια συνδεδεμένης λίστας πλεονεκτεί σε σχέση με την υλοποίηση με χρήση πίνακα διότι:

1. Στη συνδεδεμένη λίστα μπορούμε να εισάγουμε και να διαγράφουμε στοιχεία σε οποιοδήποτε σημείο
2. Δεν αντιμετωπίζουμε το πρόβλημα της υπερχείλισης
3. Παρόλο που δεν λύνουμε το πρόβλημα της υπερχείλισης δεν αντιμετωπίζουμε το πρόβλημα της εικονικής υπερχείλισης
4. Στη συνδεδεμένη λίστα μπορούμε να εισάγουμε στοιχεία οποιοδήποτε τύπου δεδομένων.

ΔΕΝΤΡΑ

Q18. Η διαδρομή από τη ρίζα ενός δέντρου προς έναν κόμβο του είναι:

1. Μοναδική
2. Υπάρχουν περισσότερες από μία διαδρομές
3. Μοναδική μόνο στην περίπτωση που το δέντρο είναι δυαδικό
4. Εξαρτάται από το επίπεδο που βρίσκεται ο κόμβος

Q19. Σαν ***ύψος (heigth)*** ενός κόμβου σε ένα δένδρο ορίζουμε;

1. Τον αριθμό του τελευταίου επιπέδου μείον την απόστασή του από τη ρίζα.
2. Τον αριθμό των κόμβων που υπάρχουν στο επίπεδο που βρίσκεται ο κόμβος
3. τον αριθμό του επιπέδου στο οποίο βρίσκεται ο κόμβος αυτός.
4. τον αριθμό των υποδέντρων του συγκεκριμένου κόμβου

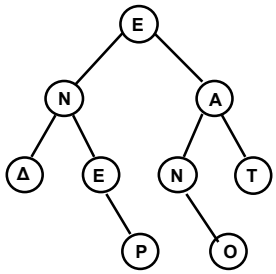
Q20. Δυαδικό δέντρο αναζήτησης *(binary search tree)* είναι αυτό του οποίου:

1. Τα δεδομένα είναι τοποθετημένα σε αύξουσα τάξη.
2. Τα δεδομένα είναι τοποθετημένα με τη βοήθεια του αλγόριθμου inorder
3. Οι τιμές που βρίσκονται στο αριστερό υποδέντρο είναι μικρότερες από την τιμή της ρίζας και οι τιμές που βρίσκονται στο δεξιό υποδέντρο είναι μεγαλύτερες.
4. Για οποιονδήποτε κόμβο όλες οι τιμές που βρίσκονται στο αριστερό υποδέντρο του είναι μικρότερες από την τιμή του κόμβου και όλες οι τιμές που βρίσκονται στο δεξιό υποδέντρο είναι μεγαλύτερες.

Q21. Σωρός (*heap*) είναι:

1. Η δομή δεδομένων στην οποία τα στοιχεία έχουν τοποθετηθεί με διάταξη τύπου LIFO.
2. το δυαδικό δέντρο που είναι σχεδόν πλήρες και οι τιμές των κόμβων είναι τοποθετημένες με τέτοιο τρόπο ώστε ο κάθε κόμβος πατέρας να έχει μεγαλύτερη τιμή από τα παιδιά του.
3. το δυαδικό δέντρο που είναι σχεδόν πλήρες και οι τιμές των κόμβων κάθε επιπέδου είναι μικρότερες από αυτές των κόμβων του προηγούμενου επιπέδου.
4. το δυαδικό δέντρο που είναι πλήρως ισοζυγισμένο και οι τιμές των κόμβων είναι τοποθετημένες με τέτοιο τρόπο ώστε ο κάθε κόμβος πατέρας να έχει μεγαλύτερη τιμή από τα παιδιά του.

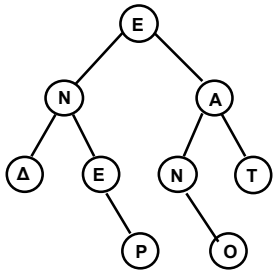
Q22. Δίνεται το παρακάτω δυαδικό δέντρο:



Αν εφαρμόσουμε τον **ενθεματικό (*inorder*)** τρόπο διέλευσης η σειρά επίσκεψης των κόμβων θα είναι:

1. N E A Δ E P N T O
2. Δ N E P E O N A T
3. Δ N E P E N O A T
4. P Δ E N E O N A T

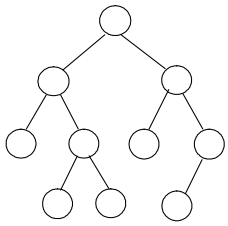
Q23. Δίνεται το παρακάτω δυαδικό δέντρο:



Αν εφαρμόσουμε τον **προθεματικό (*preorder*)** τρόπο διέλευσης η σειρά επίσκεψης των κόμβων θα είναι:

1. E N A Δ E N T P O
2. E N Δ E P A N T O
3. P E N Δ E N A T O
4. E N Δ E P A N T O

Q24. Δίνεται το παρακάτω δυαδικό δέντρο το οποίο περιέχει δέκα (10) κόμβους με τη συγκεκριμένη δομή:



Με ποια σειρά πρέπει να βρίσκονται οι παρακάτω αριθμοί κατά την είσοδό τους, ώστε το δέντρο που θα προκύψει να είναι **δυαδικό δέντρο αναζήτησης (*binary search tree*)**: **10 20 30 40 50 60 70 80 90 99**

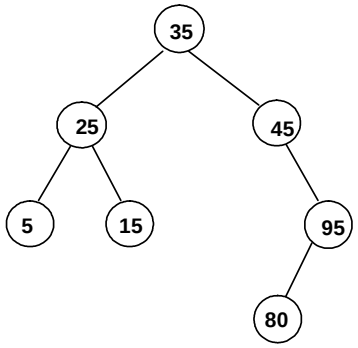
1. **99 90 80 70 60 50 40 30 20 10**
2. **60 10 40 20 30 50 80 70 90 99**
3. **60 80 20 10 70 40 50 30 99 90**
4. **60 50 40 30 20 10 99 90 80 70**

Q25. Αν οι παρακάτω αριθμοί εισαχθούν σε ένα δυαδικό δέντρο αναζήτησης

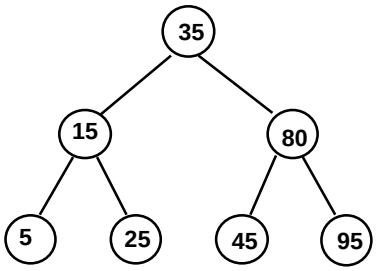
35 15 25 80 95 5 45

αυτό θα έχει την εξής μορφή:

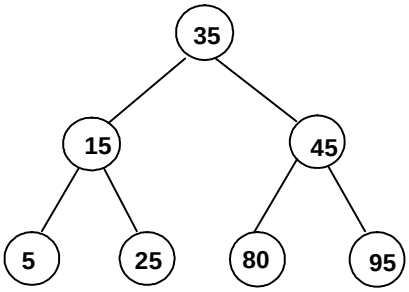
1.



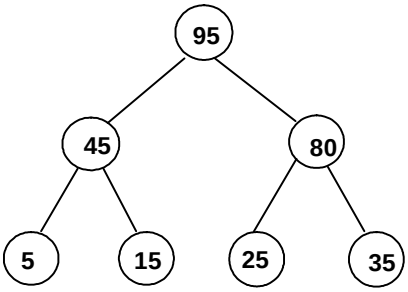
2.



3.



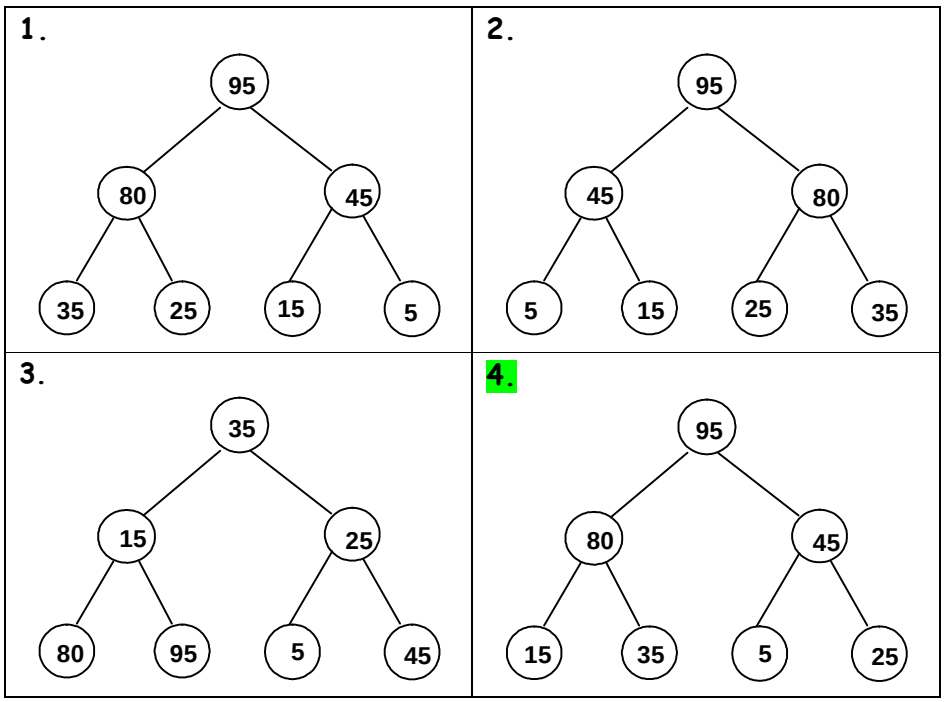
4.



Q26. Αν οι παρακάτω αριθμοί εισαχθούν σε ένα σωρό (heap)

35 15 25 80 95 5 45

αυτός θα έχει την εξής μορφή:



ΑΡΧΕΙΑ

Q27. Η έννοια του **Ρεύματος (*Stream*)** στη Java αναφέρεται:

1. Σε ένα αρχείο της Java το οποίο μπορεί να είναι είτε αρχείο εισόδου είτε αρχείο εξόδου.
2. Σε ένα πίνακα στη κεντρική μνήμη ο οποίος φιλοξενεί τα δεδομένα με την ίδια σειρά που υπάρχουν σε ένα αρχείο.
3. Σε ένα αντικείμενο το οποίο αναπαριστά μία σειριακή ροή δεδομένων από μία πηγή προς έναν προορισμό.
4. Σε κανένα από τα παραπάνω

Q28. Η έννοια του **Φίλτρου (*Filter*)** στη Java αναφέρεται:

1. Σε ένα ρεύμα **(stream)** που φιλτράρει δεδομένα όπως αυτά διαβάζονται ή γράφονται στο ρεύμα
2. Ένα ειδικό πρόγραμμα που επεξεργάζεται τα δεδομένα ενός ρεύματος.
3. Μια ενσωματωμένη δομή δεδομένων της Java που επεξεργάζεται τα δεδομένα ενός ρεύματος.
4. Σε κανένα από τα παραπάνω

Q29. Η κλάση **File** της Java έχει δημιουργηθεί:

1. Για την επεξεργασία των αρχείων από Bytes
2. Για την επεξεργασία των αρχείων κειμένου
3. Για την επεξεργασία των αρχείων τυχαίας πρόσβασης
4. Για κανένα από τους παραπάνω λόγους

Q30. Ένα αρχείο **τυχαίας πρόσβασης (*random access file*)** διαφέρει από ένα ακολουθιακό αρχείο **(*sequential file*):**

1. Από πλευράς χειρισμού ενός αρχείου δεν υπάρχει καμμία διαφοροποίηση ανάμεσα στις δύο κατηγορίες
2. Στο αρχείο τυχαίας πρόσβασης τα δεδομένα μπορούν να τοποθετηθούν ταξινομημένα ενώ στο ακολουθιακό αρχείο όχι
3. Ένα αρχείο τυχαίας πρόσβασης μπορεί να είναι ταυτόχρονα και αρχείο εισόδου και αρχείο εξόδου ενώ ένα ακολουθιακό αρχείο όχι
4. Το ακολουθιακό αρχείο μπορεί να χρησιμοποιηθεί για την σειριακή επεξεργασία χαρακτήρων ενώ το αρχείο τυχαίας πρόσβασης όχι