METROPOLITAN COLLEGE

University of East London

CN5005 Data Structures and Algorithms

Εξαμηνιαία Ομαδική Εργασία (Ακαδημαϊκό Έτος 2025 - 2026)

Ομαδική εργασία:	**έως 3 φοιτητές**
Συνολικό βάρος:	**50 % του μαθήματος**
Τελική παράδοση:	**Εβδομάδα 12**
Viva (προφορική εξέταση):	**Εβδομάδα 13 (χωρίς διαδίκτυο)**

Σκοπός και γενικές οδηγίες

Η εργασία αποσκοπεί να σας εξοικειώσει με:

- την υλοποίηση και κατανόηση **δυαδικών δέντρων αναζήτησης (BST)** και ισορροπημένων δέντρων **AVL**,
- την επεξεργασία δομημένων δεδομένων (αρχεία CSV) για την αναγνώριση σχέσεων,
- και τη σύνθεση ενός πλήρως λειτουργικού προγράμματος με κατάλληλη τεκμηρίωση και δοκιμές.

Η εργασία ολοκληρώνεται ως ένα ενιαίο έργο που υποβάλλεται στην Εβδομάδα 12 και παρουσιάζεται στη Viva της Εβδομάδας 13.

Η υλοποίηση πρέπει να γίνει αποκλειστικά σε Java (έκδοση 17+).

Κάθε ομάδα παραδίδει ένα μοναδικό αρχείο .zip με όλα τα παραδοτέα που περιγράφονται παρακάτω.

Δομή και βαθμολογική κατανομή

Ενότητα	Περιγραφή	Μονάδες
A. Δυαδικό Δέντρο (BST)	**Κόμβοι - εισαγωγή - διαγραφή - διασχίσεις**	**25**
B. AVL Εξισορρόπηση & Αλλαγή Κλειδιού	**Περιστροφές - ισορροπία - changeKey**	**15**
C. Ανάγνωση & Έλεγχος persons.csv	**Δημιουργία/φόρτωση dataset - parser - tests**	**10**
D. Αναγνώριση Συγγενικών Σχέσεων	**Πατέρας/Μητέρα - Αδέλφια - Ξαδέρφια - CLI**	**25**
E. Σύνθετες Σχέσεις	**Half-siblings - In-law - Spouse (2 επιλογές)**	**5**
F. Unit Tests - Screencast - Viva	**Δοκιμές - παρουσίαση - εξέταση στη τάξη**	**20**
Σύνολο		**100**

Οδηγίες υλοποίησης και βαθμολογικά κριτήρια

Η τελική υποβολή πρέπει να είναι ένα συμπιεσμένο αρχείο .zip που περιέχει, στον ριζικό του κατάλογο: φάκελο src/ με όλο τον πηγαίο κώδικα, φάκελο tests/ με όλα τα unit tests, το αρχείο persons.csv (ή ένδειξη χρήσης του παρεχόμενου), το screencast (ή σύνδεσμο), και ένα README.md με οδηγίες εκτέλεσης και τα στοιχεία της ομάδας (ονόματα, ΑΜ, ποιος έκανε τι). Η δομή και ονομασία φακέλων πρέπει να είναι σαφής και εκτελέσιμη χωρίς επιπλέον ρυθμίσεις.

Μέρος Α — Υλοποίηση Δυαδικού Δέντρου (Σύνολο 25)

Στο πρώτο μέρος θα υλοποιήσετε από την αρχή ένα Binary Search Tree (BST) με δυνατότητα καταγραφής διπλοτύπων και στη συνέχεια θα το επεκτείνετε ώστε να λειτουργεί ως AVL (ισορροπημένο δέντρο).

Σημείωση για τα δεδομένα εισαγωγής: Κάθε ομάδα παράγει τα εισαγόμενα κλειδιά ως εξής: κάθε μέλος παίρνει τα **τέσσερα τελευταία ψηφία** του Αριθμού Μητρώου (ΑΜ). Τα μέλη ταξινομούνται αλφαβητικά κατά επώνυμο και οι τετραψήφιες τιμές εισάγονται με αυτή τη

σειρά. Αν δεν έχετε 6 τιμές, επαναλάβετε κυκλικά μέχρι να έχετε τουλάχιστον **6 εισαγωγές**. Διπλότυπα κρατιούνται και δοκιμάζουν τη λογική count.

1.1 Κλάση Node — Node class (5 μονάδες)

Υλοποιήστε κλάση Node που θα περιέχει τα πεδία: key:int, count:int, left:Node, right:Node. Ο constructor πρέπει να δέχεται το key, να θέτει count = 1 και να αρχικοποιεί τις αναφορές παιδιών σε null. Στην τεκμηρίωση του αρχείου σας εξηγήστε σύντομα γιατί χρειάζεται το count και πώς το χρησιμοποιείτε.

1.2 Μέθοδος εισαγωγής — insert (8 μονάδες)

Υλοποιήστε τη μέθοδο insert(int key) που εισάγει σωστά ένα κλειδί στο BST. Η μέθοδος ξεκινά από τη ρίζα και προχωρά αριστερά ή δεξιά ανάλογα με τη σχέση key < node.key ή key > node.key. Αν βρεθεί κόμβος με ίδιο key, δεν δημιουργεί νέο κόμβο· αυξάνει απλώς το count του υπάρχοντος. Στις παραδοχές/σημειώσεις του κώδικα αναφέρετε τι θα συμβεί αν εισάγετε null ή μη-ακέραιες τιμές.

1.3 Μέθοδος διαγραφής — delete (4 μονάδες)

Υλοποιήστε delete(int key) που αφαιρεί το κλειδί από το δέντρο. Η υλοποίηση πρέπει να καλύπτει τις τρεις κλασικές περιπτώσεις: κόμβος φύλλο, κόμβος με ένα παιδί, κόμβος με δύο παιδιά. Στην τελευταία περίπτωση αντικαταστήστε το κλειδί του κόμβου από τον inorder διάδοχο (ή predecessor) και διαγράψτε τον διάδοχο. Αν στο node υπάρχει count > 1, η delete μπορεί να μειώνει απλώς τον count κατά 1 (νεότερη επιλογή) — διευκρινίστε στην αναφορά ποια πολιτική εφαρμόσατε.

1.4 Διασχίσεις και επίδειξη — traversals & demo (8 μονάδες)

Υλοποιήστε preorder(), inorder() και postorder() που επιστρέφουν σειρές/λίστες με τα κλειδιά ή tuples (key,count). Δημιουργήστε ένα main πρόγραμμα ή script που: κατασκευάζει το δέντρο με τις αριθμητικές εισαγωγές από τους ΑΜ της ομάδας, εκτελεί και εμφανίζει τις τρεις διασχίσεις και περιλαμβάνει μικρό log που εξηγεί την έξοδο. Η inorder() πρέπει να εμφανίζει τα κλειδιά σε αύξουσα σειρά (λαμβάνοντας υπ’ όψη και το count).

Μέρος Β — AVL εξισορρόπηση και αλλαγή κλειδιού (Σύνολο 15)

Μετά την επιτυχή υλοποίηση του BST, επεκτείνετε τον κώδικά σας ώστε να διατηρεί ισορροπία αυτόματα (AVL).

2.1 Εξισορρόπηση AVL — balancing (10 μονάδες)

Προσθέστε στο Node πεδίο height:int. Μετά από κάθε insert ή delete υπολογίστε/ενημερώστε τα ύψη και τον balance factor height(left) − height(right). Όταν η απόκλιση υπερβαίνει το 1, εφαρμόστε την κατάλληλη περιστροφή: LL, RR, LR, RL. Στο σχόλιο/README περιγράψτε σε ποιες περιπτώσεις εμφανίζεται κάθε περιστροφή και ένα παράδειγμα εκτέλεσης (μία σύντομη trace).

2.2 Αλλαγή κλειδιού — changeKey (5 μονάδες)

Υλοποιήστε changeKey(int oldKey, int newKey). Η απλούστερη και αποδεκτή λύση είναι: αναζητείτε το node με oldKey, αποθηκεύετε το count, καλείτε delete(oldKey) (ή μειώνετε το count) και μετά insert(newKey) με τον ίδιο count. Εναλλακτικά, μπορείτε να αλλάξετε απευθείας το key στον κόμβο και να επανατοποθετήσετε τον κόμβο εφαρμόζοντας rotations όταν χρειάζεται — αν ακολουθήσετε αυτή τη δεύτερη προσέγγιση, τεκμηριώστε την στην αναφορά. Κατά την viva θα ζητηθεί να εξηγήσετε την επιλογή σας.

Μέρος C — persons.csv: δημιουργία/χρήση dataset και ανάγνωση (Σύνολο 10)

Το persons.csv είναι το κεντρικό dataset για όλο το γενεαλογικό μέρος. Αυτό το κομμάτι περιγράφει **ακριβώς** τι είναι το αρχείο, πού μπαίνει, πώς το φτιάχνετε / το χρησιμοποιείτε, και ποιοι έλεγχοι ζητούνται.

C.1 Επιλογές για το persons.csv (4 μονάδες)

Η ομάδα επιλέγει **μία** από τις παρακάτω επιλογές:

- **Επιλογή Α — Χρήση παρεχόμενου αρχείου:** Ο διδάσκων παρέχει ένα persons.csv με 25 εγγραφές σε 3 γενιές. Η ομάδα χρησιμοποιεί αυτό το αρχείο και συνεχίζει απευθείας στη C.2 και στο Μέρος D.
- **Επιλογή Β — Δημιουργία δικού σας αρχείου:** Η ομάδα δημιουργεί persons.csv με **τουλάχιστον 25 άτομα** και **ελάχιστες 3 γενιές**. Συνιστάται να βασιστείτε σε μια

συνεκτική οικογένεια (φανταστική ή ιστορική) ώστε οι σχέσεις να έχουν νόημα. Το αρχείο παραδίδεται μαζί με την τελική υποβολή.

Σημείωση: Αν επιλέξετε να δημιουργήσετε το δικό σας dataset, προσθέστε στο REFLECTION.md (Μέρος CW2) μια παράγραφο που εξηγεί την προέλευση/συνάφεια των στοιχείων (DATA_PROVENANCE).

C.2 Μορφή αρχείου (Υποχρεωτικό βήμα χωρίς βαθμολόγηση)

Το αρχείο persons.csv πρέπει να είναι UTF-8, με header και κόμμα ως διαχωριστικό. Η πρώτη γραμμή πρέπει να είναι:

id,name,gender,father_id,mother_id,spouse_id

Περιγραφές πεδίων:

- id: μοναδικός αναγνωριστικός (προτείνεται θετικός ακέραιος ή string χωρίς κενά).
- name: πλήρες όνομα (π.χ. Μαρία Παπαδοπούλου).
- gender: Male / Female / Unknown.
- father_id, mother_id: αναφορά στο id του γονέα (κενό αν άγνωστο).
- spouse_id: αναφορά στο id του/της συζύγου (κενό αν άγνωστο ή αν δεν υπάρχει).

Απαγορεύεται να παραδίδετε το αρχείο σε άλλη μορφή (π.χ. Excel .xlsx) — **μόνο CSV** με header. Ονομασία αρχείου **πρέπει** να είναι persons.csv.

C.3 Φόρτωση στη μνήμη και δομές (3,5 μονάδες)

Υλοποιήστε parser που:

1. Διαβάζει το persons.csv.
2. Για κάθε γραμμή δημιουργεί αντικείμενο Person με τα πεδία (id, name, gender, father_id, mother_id, spouse_id).
3. Αποθηκεύει τα Person σε δύο ευρετήρια: έναν χάρτη id → Person και έναν δεύτερο name → id (για εύκολη αναζήτηση με όνομα).
4. Μετά τη φόρτωση, το πρόγραμμα εκτυπώνει όλα τα ονόματα μαζί με το φύλο τους (μία γραμμή ανά πρόσωπο) ώστε να επιβεβαιωθεί ο επιτυχής parse.

Συμπεριλάβετε κατάλληλο logging/μηνύματα σφαλμάτων για περιπτώσεις λανθασμένων ή ελλειπουσών γραμμών.

C.4 Έλεγχοι ορθότητας (2,5 μονάδες)

Προσθέστε 2-4 μικρά unit tests που επαληθεύουν:

- ότι ο συνολικός αριθμός εγγραφών που φορτώθηκαν ισούται με τις γραμμές του CSV (εκτός της κεφαλίδας),
- ότι συγκεκριμένα id αντιστοιχούν στα αναμενόμενα ονόματα,
- ότι κενά πεδία (father_id, mother_id, spouse_id) αποθηκεύονται ως null ή κενό string (σύμφωνα με την υλοποίησή σας).

Τεκμηριώστε τα tests στο README και τρέξτε τα με μία εντολή (π.χ. mvn test(JUnit)).

Μέρος D — Ανίχνευση Συγγενειών και queries (Σύνολο 25)

Μετά τη φόρτωση του persons.csv θα αναπτύξετε συναρτήσεις που ελέγχουν οικογενειακές σχέσεις. Οι μέθοδοι πρέπει να είναι καθαρές, καλά τεκμηριωμένες και να έχουν unit tests.

D.1 Πατέρας / Μητέρα (4 μονάδες)

Υλοποιήστε isFather(idA, idB) και isMother(idA, idB) που ελέγχουν αν το άτομο με idA είναι ο πατέρας ή η μητέρα του idB. Οι μέθοδοι επιστρέφουν boolean και πρέπει να διαχειρίζονται περιπτώσεις όπου κάποιο από τα ids δεν υπάρχει (στο οποίο πρέπει να επιστρέφεται false και να γίνεται κατάλληλο log/warning).

D.2 Παιδί / Αδέλφια (4 μονάδες)

Υλοποιήστε isChild(idA, idB) (αν το A είναι παιδί του B) και isSibling(idA, idB) (αν δύο άτομα είναι αδέλφια — τουλάχιστον ένας κοινός γονέας). Προσέξτε την περίπτωση του ιδίου προσώπου (idA == idB) και χειριστείτε την κατάλληλα.

D.3 Παππούς/Γιαγιά και Εγγόνια (4 μονάδες)

Υλοποιήστε isGrandparent(idA, idB) και isGrandchild(idA, idB). Η λογική: Α είναι παππούς/γιαγιά του Β αν υπάρχει γονέας του Β του οποίου ο γονέας είναι ο Α.

D.4 Πρώτα Ξαδέρφια (4 μονάδες)

Υλοποιήστε isFirstCousin(idA, idB) που επιστρέφει true αν ο πατέρας ή η μητέρα του Α είναι αδελφός/αδελφή του πατέρα ή της μητέρας του Β.

D.5 Συνάρτηση relation(nameA, nameB) και CLI (9 μονάδες)

Υλοποιήστε υψηλού επιπέδου συνάρτηση relation(nameA, nameB) που:

1. Αναζητά τα ids μέσω του name → id ευρετηρίου.
2. Καλεί διαδοχικά τις μεθόδους isFather, isMother, isChild, isSibling, isGrandparent, isGrandchild, isFirstCousin και επιστρέφει την **πρώτη** σχέση που βρέθηκε σε ανθρώπινη μορφή (π.χ. «Η Μαρία Παπαδοπούλου είναι μητέρα του Γιάννη Ιωάννου»).
3. Αν καμία σχέση δεν βρεθεί, επιστρέφει «Δεν σχετίζονται».

Προσφέρετε και ένα απλό CLI: εντολή relation "Όνομα1" "Όνομα2" για χρήση από τερματικό. Η υλοποίηση του CLI πρέπει να γίνει σε Java, και το πρόγραμμα να εκτελείται μέσω Java runtime (π.χ. java -jar ...). Το CLI πρέπει να αντιμετωπίζει περιπτώσεις ονόματος που δεν υπάρχει και να επιστρέφει σαφές μήνυμα προς τον χρήστη.

Μέρος Ε — Σύνθετες σχέσεις (Σύνολο 5)

Επιλέξτε **δύο** από τις παρακάτω και υλοποιήστε τις (2.5 μονάδες η καθεμία), και ενσωματώστε τον έλεγχο στη relation:

1. **Half-siblings (ετεροθαλή αδέλφια):** ίδια μόνο ο ένας γονέας.
2. **In-law relationships (σχέσεις εξ αγχιστείας):** π.χ. πεθερός/πεθερά = γονέας του/της συζύγου.
3. **Spouse check (σύζυγοι):** spouse_id πεδίου.

Στην αναφορά REFLECTION.md εξηγήστε τις θεώρησεις και τις υποθέσεις που κάνατε (π.χ. μονογονεϊκές οικογένειες, πολλαπλοί γάμοι).

Μέρος F — Unit Tests, Screencast και Υποχρεωτικό Viva (Σύνολο 20)

F.1 Unit Tests (8 μον.)

Συγκεντρώστε όλα τα unit tests στο φάκελο tests/ και βεβαιωθείτε ότι τρέχουν με μία εντολή. Όλα σε φάκελο tests/, εκτελούνται με μία εντολή (π.χ. mvn test με χρήση JUnit).

Το README.md πρέπει να περιλαμβάνει οδηγίες για build και εκτέλεση με Maven, καθώς και παραδείγματα CLI εκτέλεσης με Java (π.χ. java -jar cli.jar relation \"name1\" \"name2\").

F.2 Screencast (6 μον.)

Δημιουργήστε ένα screencast 1-2 λεπτών που:

- δείχνει γρήγορα την εισαγωγή και διαγραφή κόμβων,
- εμφανίζει τις τρεις διασχίσεις,
- δείχνει φόρτωση του persons.csv και τρεις κλήσεις relation (μία άμεση σχέση, μία ξαδέρφων, μία «Δεν σχετίζονται»).

F.3 Viva (6 μον.)

Εκτέλεση ζωντανά (changeKey, insert/delete, relation) και ερωτήσεις πολυπλοκότητας.

Κατά τη viva θα κληθείτε να εκτελέσετε ζωντανά: changeKey, μια σειρά από insert/delete και ένα query relation. Προετοιμαστείτε επίσης να απαντήσετε σε ερωτήσεις σχετικά με πολυπλοκότητα (π.χ. O(log n) για AVL) και επιλογές υλοποίησης.