ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
Τμήμα Πληροφορικής και Τηλεπικοινωνιών
1η Εργασία - Τμήμα: Άρτιων Αριθμών Μητρώου
K24: Systems Programming – Εαρινό Εξάμηνο ‘20

Κωστόπουλος Σταύρος
Α.Μ.: 1115201700068

Η υλοποίηση του προγράμματος της 1ης Εργασίας, προκύπτει από την σύνθεση εφτά (7) πηγαίων αρχείων της μορφής .c (Το πρόγραμμα αναπτύχθηκε στην γλώσσα προγραμματισμού C).Τα εφτά αυτά αρχεία είναι τα παρακάτω:

- main.c
- functions.c
- hashtable.c
- kouvades.c
- queries.c
- tree.c
- topk.c

Παρακάτω ακολουθούν μερικές σημειώσεις που επεξηγούν την υλοποίηση της εργασίας:

1. Ξεκινώντας,το πρόγραμμα ανοίγει το αρχείο patientRecordsFile και διαβάζοντας μία μία τις εγγραφές τις αποθηκεύει σε μία λίστα κόμβων τύπου patient,όπου κάθε κόμβος περιέχει το recordID, patientFirstName, patientLastName, diseaseID, country, entryDate, exitDate ενός ασθενή.(βλέπε mylib.h και functions.c)

2. Τα entryDates και exitDates αποθηκεύονται και ο χειρισμός τους επιτυγχάνεται μέσω μιας δομής myDate (βλέπε mylib.h) και κατάλληλων συναρτήσεων που έχουν υλοποιηθεί μέσα στο πηγαίο αρχείο functions.c .

3. Όπως ζητείται στην εκφώνηση της εργασίας πριν από κάθε εγγραφή στην λίστα με κόμβους τύπου patient γίνεται έλεγχος για διπλότυπα recordID.Ο έλεγχος αυτός θα ήταν εξαιρετικά χρονοβόρος αν γινόταν με traverse της λίστας κάθε φορά που θέλαμε να προσθέσουμε μια νέα εγγραφή σε αυτήν.Για να λυθεί αυτό,υλοποιήθηκε άλλος ένας πίνακας κατακερματισμού(hash table),στον οποίο χρησιμοποιούνται κουβάδες ακριβώς με τον ίδιο τρόπο που χρησιμοποιούνται για τα disease και country Hash tables.Η δέσμευση μνήμης για το recordID hash table γίνεται δυναμικά εννοείται,και το μέγεθος του καθορίζεται μέσω μια define μέσα στο αρχείο τύπου .h με όνομα kouvas.h (#define RECORDHASHENTRIES).

4. Τα binary search trees που χρησιμοποιούνται για κάθε country και diseaseID που γίνονται hashed είναι τύπου Red Black Tree που είχαν υλοποιηθεί από εμένα στην 1η εργασία του μαθήματος Κ22-Λειτουργικά Συστήματα(Τμήμα Αρτίων) το προηγούμενο εξάμηνο.Υπάρχει πλήρης κατανόηση για τα δέντρα αυτά και προσαρμόστηκαν ανάλογα με τα δεδομένα της άσκησης.(βλέπε dentro.h και tree.c)

5. Η συνάρτηση κατακερματισμού που χρησιμοποιήθηκε και για τα τρία hash tables(recordID, disease, country),προέρχεται από δημόσια πηγή στον παγκόσμιο ιστό,ύστερα από ερώτηση στο piazza.(url: http://www.cse.yorku.ca/~oz/hash.html)

6. Τέλος ύστερα από πλήρη κατανόηση,χρησιμοποιήθηκαν οι συναρτήσεις preorder(), max_heapify() από δημόσια πηγή και ύστερα προσαρμόστηκαν στα δεδομένα της άσκησης για την επίλυση των εντολών topk-Diseases και topk-Countries(url: https://stackoverflow.com/questions/24801613/max-heapify-a-binary-tree)

7. Το παραδοτέο πρόγραμμα,ικανοποιεί όλες τις απαιτήσεις και τις εντολές της άσκησης,χωρίς memory leaks και errors κατά τον τερματισμό του.

8. Τέλος,μια ενδεικτική εκτέλεση του προγράμματος που παραδόθηκε είναι η εξής:

    $ make
    $ ./diseaseMonitor -p patientsFileName -h1 10 -h2 10 -b 80
    $ make clean

    Και μία ενδεικτική εκτέλεση με την χρήση του validator script,είναι η εξής:

    $ ./validator.sh small.txt 10 10 80
    $ make clean