Εισαγωγή

Στην εργασία αυτή θα υλοποιήσετε ένα πρόγραμμα που θα δέχεται, θα επεξεργάζεται, θα καταγράφει και θα απαντάει ερωτήματα για κρούσματα ιώσεων. Συγκεκριμένα θα υλοποιήσετε ένα σύνολο δομών (hash tables, linked lists, binary trees) που επιτρέπουν την εισαγωγή και επερωτήσεις σε μεγάλο όγκο εγγραφών τύπου `patientRecord`. Αν και τα δεδομένα της άσκησης θα προέρχονται από αρχεία, τελικά όλες οι εγγραφές θα αποθηκεύονται μόνο στην κύρια μνήμη.

Interface της εφαρμογής

Η εφαρμογή θα ονομάζεται `diseaseMonitor` και θα χρησιμοποιείται ως εξής:

```
./diseaseMonitor -p patientRecordsFile -h1 diseaseHashtableNumOfEntries
                  -h2 countryHashtableNumOfEntries -b bucketSize
```

όπου:

- Η παράμετρος `diseaseHashtableNumOfEntries` είναι ο αριθμός θέσεων ενός πίνακα κατακερματισμού που θα κρατάει η εφαρμογή για τον εντοπισμό πληροφοριών για τους ασθενείς.
- Η παράμετρος `countryHashtableNumOfEntries` είναι ο αριθμός θέσεων ενός πίνακα κατακερματισμού που θα κρατάει η εφαρμογή για τον εντοπισμό πληροφοριών για κρούσματα ανά χώρα.
- Η παράμετρος `bucketSize` είναι ο αριθμός των Bytes που δίνει το μέγεθος του κάθε bucket στους πίνακες κατακερματισμού.
- Το `patientRecordsFile` (ή κάποιο άλλο όνομα αρχείου) είναι ένα αρχείο που περιέχει μια σειρά από εγγραφές ασθενών προς επεξεργασία. Κάθε γραμμή του αρχείου αυτού περιγράφει ένα κρούσμα μιας ίωσης, το όνομα του ασθενή, σε ποια χώρα, την ημερομηνία που εισήχθη σε νοσοκομείο και την ημερομηνία που πήρε εξιτήριο. Για παράδειγμα αν τα περιεχόμενα του αρχείου είναι:

```
889 Mary Smith COVID-2019 China 25-1-2019 27-1-2019
776 Larry Jones SARS-1 Italy 10-02-2003 -
125 Jon Dupont  H1N1 USA 12-02-2016 15-02-2016
```

σημαίνει πως έχουμε τρεις εγγραφές που περιγράφουν τρία κρούσματα σε τρεις διαφορετικές χώρες (Κίνα, Ιταλία, ΗΠΑ). Στη δεύτερη εγγραφή, δεν υπάρχει ημερομηνία εξιτηρίου (ο ασθενής παραμένει στο νοσοκομείο). Συγκεκριμένα, μια εγγραφή είναι μια γραμμή ASCII κειμένου που αποτελείται από τα εξής στοιχεία:

1. `recordID`: μια συμβολοσειρά (μπορεί να έχει και ένα μόνο ψηφίο) που με μοναδικό τρόπο καθορίζει την κάθε τέτοια εγγραφή.
2. `patientFirstName`: μια συμβολοσειρά που αποτελείται από γράμματα χωρίς κενά.
3. `patientLastName`: μια συμβολοσειρά που αποτελείται από γράμματα χωρίς κενά.
4. `diseaseID`: μια συμβολοσειρά που αποτελείται από γράμματα, αριθμούς, και ενδεχομένως και μια παύλα "-" αλλά χωρίς κενά.
5. `country`: μια συμβολοσειρά που αποτελείται από γράμματα χωρίς κενά.
6. `entryDate`: ημερομηνία που γίνεται εισαγωγή στο νοσοκομείο. Πρέπει να έχει την μορφή DD-MM-YYYY όπου το DD εκφράζει την ημέρα, το MM το μήνα, και το YYYY το χρόνο της εισαγωγής του ασθενή.
7. `exitDate`: ημερομηνία που βγήκε ο ασθενής από το νοσοκομείο. Πρέπει να έχει την μορφή DD-MM-YYYY όπου το DD εκφράζει την ημέρα, το MM το μήνα, και το YYYY το χρόνο της εξόδου του ασθενή ή παύλα (-) που σημαίνει πως ο ασθενής δεν έχει πάρει ακόμα εξιτήριο.

Ξεκινώντας, η εφαρμογή σας θα πρέπει να ανοίξει το αρχείο `patientRecordsFile` να διαβάσει μία-μία τις γραμμές και να αρχικοποιήσει και να αποθηκεύσει στη μνήμη τις δομές δεδομένων που θα χρησιμοποιεί κατά την εκτέλεση ερωτημάτων. Θα πρέπει να ελέγχετε πως τα στοιχεία στα αρχεία είναι έγκυρα. Για παράδειγμα, αν στο αρχείο `patientRecordsFile`, υπάρχουν δύο εγγραφές με

το ίδιο recordID, θα πρέπει να χειριστείτε το λάθος παρουσιάζοντας το κατάλληλο μήνυμα και βγαίνοντας από την εφαρμογή. Επίσης, αν εντοπίσετε μια εγγραφή όπου η entryDate είναι μεταγενέστερη της exitDate, τότε η εφαρμογή θα πρέπει να παρουσιάσει ένα μήνυμα πως η εγγραφή απορρίπτεται.

Όταν η εφαρμογή τελειώσει την επεξεργασία του patientRecordsFile αρχείου, θα περιμένει είσοδο από τον χρήστη από το πληκτρολόγιο. Ο χρήστης θα μπορεί να δίνει τις ακόλουθες εντολές (τα ορίσματα σε [ ] είναι προαιρετικά):

- /globalDiseaseStats [date1 date2]

Η εφαρμογή θα τυπώνει για κάθε ίωση, τον αριθμό κρουσμάτων που έχουν καταγραφεί στο σύστημα. Αν δοθούν date1 date2 τότε η εφαρμογή θα τυπώνει για κάθε ίωση, τον αριθμό κρουσμάτων που έχουν καταγραφεί στο σύστημα μέσα στον χρονικό διάστημα [date1...date2].
Εάν υπάρχει ο ορισμός για [date1] θα πρέπει να υπάρχει και ορισμός για [date2], αλλιώς, θα τυπώνεται μήνυμα λάθους στον χρήστη.

- /diseaseFrequency virusName [country] date1 date2

Αν δεν δοθεί country όρισμα, η εφαρμογή θα τυπώνει για την ασθένεια virusName τον αριθμό κρουσμάτων που έχουν καταγραφεί στο σύστημα μέσα στο διάστημα [date1...date2]. Αν δοθεί country όρισμα, η εφαρμογή θα τυπώνει για την ασθένεια virusName, τον αριθμό κρουσμάτων στην χώρα country που έχουν καταγραφεί μέσα στο διάστημα [date1...date2].

- /topk-Diseases k country [date1 date2]

Η εφαρμογή θα τυπώνει, για τη χώρα country, τις ασθένειες που αποτελούν το top k των κρουσμάτων στο διάστημα [date1...date2] εφόσον δοθεί. Εάν υπάρχει ο ορισμός για [date1] θα πρέπει να υπάρχει και ορισμός για [date2], αλλιώς, θα τυπώνεται μήνυμα λάθους στον χρήστη.

- /topk-Countries k disease [date1 date2]

Η εφαρμογή θα τυπώνει, για την ίωση disease τις χώρες που έχουν εμφανίσει το top k των κρουσμάτων της συγκεκριμένης ίωσης.

- /insertPatientRecord recordID patientFirstName patientLastName diseaseID country entryDate [exitDate]

Η εφαρμογή θα εισάγει στο σύστημα μια νέα εγγραφή με τα στοιχεία της. Το στοιχείο exitDate είναι προαιρετικό.

- /recordPatientExit recordID exitDate

Η εφαρμογή θα προσθέσει το exitDate στην έγγραφη με ID recordID.

- /numCurrentPatients [disease]

Αν δοθεί το όρισμα disease, η εφαρμογή θα τυπώσει τον αριθμό ασθενών που ακόμα νοσηλεύονται με την ασθένεια disease. Αν δεν δοθεί όρισμα, η εφαρμογή θα τυπώσει για κάθε ίωση, τις έγγραφες ασθενών που ακόμα νοσηλεύονται.

- /exit

Έξοδος από την εφαρμογή. Βεβαιωθείτε πως ελευθερώνετε σωστά όλη τη δεσμευμένη μνήμη.

Δομές δεδομένων

Για την υλοποίηση της εφαρμογής μπορείτε να χρησιμοποιήσετε C ή C++. Δεν μπορείτε να χρησιμοποιήσετε όμως την Standard Template Library (STL). Όλες οι δομές δεδομένων θα πρέπει να υλοποιηθούν από εσάς.

Βεβαιωθείτε πως δεσμεύετε μόνο όση μνήμη χρειάζεται, π.χ. η ακόλουθη τακτική δε συνιστάται:

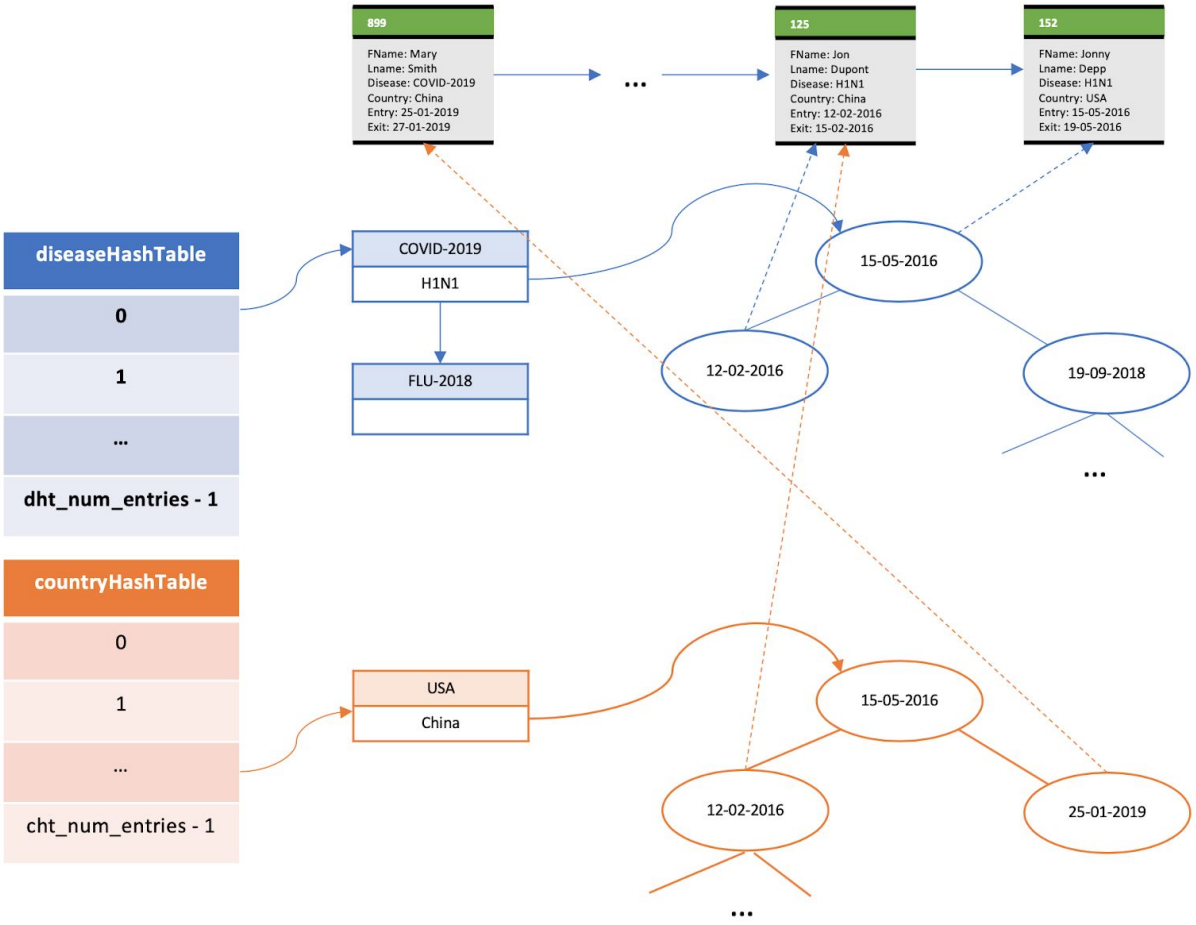
```
int diseases[512];// store up to 512 diseases, but really we don't know how many
```

Επίσης βεβαιωθείτε πως απελευθερώνετε τη μνήμη σωστά κατά την εκτέλεση του προγράμματός σας αλλά και κατά την έξοδο.

Για να ολοκληρώσετε την άσκηση θα χρειαστεί, μεταξύ άλλων, να υλοποιήσετε τις εξής δομές δεδομένων.

1. Δύο πίνακες κατακερματισμού (`diseaseHashTable` και `countryHashTable`) που με index προσφέρουν γρήγορες προσπελάσεις σε στοιχεία ασθενών ανά κρούσμα και κρουσμάτων ανά χώρα. Οι πίνακες κατακερματισμού θα χρησιμοποιούν κουβάδες για να εξυπηρετήσουν `diseases/countries` που παρουσιάζουν «σύγκρουση»/collision (δηλαδή, το αποτέλεσμα της συνάρτησης κατακερματισμού οδηγεί στο ίδιο στοιχείο του hash table). Αν χρειάζονται πιο πολλοί από ένα κουβάδες για να αποθηκευτούν δεδομένα, δημιουργούνται δυναμικά και διατάσσονται σε μια λίστα.
2. Για κάθε `disease` που γίνεται hashed σε ένα στοιχείο του `diseaseHashTable`, υπάρχει ένα σύνολο από εγγραφές ασθενών που έχουν νοσηλευτεί λόγω της ίωσης `disease`. Αυτό το σύνολο τοποθετείται σε ένα balanced binary search tree. Κάθε κόμβος του δέντρου παρέχει στοιχεία (η προσπέλαση σε στοιχεία, δείτε Σχήμα 1) μιας εγγραφής ασθενούς. Το δέντρο θα πρέπει να είναι ταξινομημένο με βάση την ημερομηνία εισαγωγής του ασθενούς στο νοσοκομείο.
3. Για κάθε `country` που γίνεται hashed σε ένα στοιχείο του `countryHashTable`, υπάρχει ένα σύνολο από εγγραφές ασθενών που έχουν νοσηλευτεί στην χώρα `country`. Αυτό το σύνολο τοποθετείται σε ένα balanced binary search tree όπου κάθε κόμβος του δέντρου παρέχει στοιχεία (η προσπέλαση σε στοιχεία) εγγραφής ασθενούς. Το δέντρο θα πρέπει να είναι ταξινομημένο με βάση την ημερομηνία εισαγωγής του ασθενούς στο νοσοκομείο.
   Επειδή θα υπάρχει επικάλυψη μεταξύ των εγγραφών στα δέντρα που προσπελάζονται μέσω `diseaseHashTable` και `countryHashTable` θα πρέπει να φροντίζετε να μην υπάρχει σπατάλη στην μνήμη, δηλαδή μια εγγραφή θα πρέπει να αποθηκεύεται μόνο μια φορά στην μνήμη και σε οποιαδήποτε δομή δεδομένων χρειάζεται πρόσβαση στην εγγραφή, η πρόσβαση θα γίνεται μέσω pointers. (δείτε Σχήμα 1 για μια πιθανή πρόταση του layout κάποιων δομών δεδομένων).
4. Για την εντολή `topk-Diseases`, το πρόγραμμα σας θα πρέπει να χτίζει on-the-fly ένα binary heap (i.e., max heap) όπου κάθε κόμβος θα κρατάει το σύνολο των κρουσμάτων μιας ίωσης και θα σας βοηθά να βρίσκετε εύκολα ποιες είναι οι ιώσεις που αποτελούν το top k των κρουσμάτων στη χώρα. Επίσης για την εντολή `topk-Countries`, το πρόγραμμά σας θα πρέπει να χτίζει on-the-fly ένα binary heap (i.e., max heap) όπου κάθε κόμβος θα κρατάει το σύνολο κρουσμάτων μιας χώρας της συγκεκριμένης ίωσης και θα σας βοήθα να βρίσκετε εύκολα ποιες είναι οι χώρες που έχουν εμφανίσει το top k των κρουσμάτων της συγκεκριμένη ίωσης.
5. Οποιαδήποτε άλλη βοηθητική δομή δεδομένων χρειαστείτε για τις ανάγκες της εργασίας.

Ο στόχος σας σε αυτή την άσκηση είναι να υπάρξει όσο το δυνατόν *μικρότερη επικάλυψη στοιχείων (data duplication)*. Βεβαιωθείτε πως για κάθε υπο-πρόβλημα που χρειάζεται να επιλύσετε κατά την υλοποίηση της άσκησης, χρησιμοποιείτε τον πιο αποτελεσματικό αλγόριθμο ή δομή δεδομένων. Όποιες σχεδιαστικές αποφάσεις και επιλογές κάνετε κατά την υλοποίηση, θα πρέπει να τις περιγράψετε στο README, στα παραδοτέα.



Σχήμα 1: Παράδειγμα Κάποιων Δομών για την εφαρμογή diseaseMonitor

Παραδοτέα

- Μια σύντομη και περιεκτική εξήγηση για τις επιλογές που έχετε κάνει στο σχεδιασμό του προγράμματός σας. 1-2 σελίδες ASCII κειμένου είναι αρκετές. Συμπεριλάβετε την εξήγηση και τις οδηγίες για το compilation και την εκτέλεση του προγράμματός σας σε ένα αρχείο README μαζί με τον κώδικα που θα υποβάλετε.
- Ο κώδικας που θα υποβάλετε θα πρέπει να είναι **δικός** σας. Απαγορεύεται η χρήση κώδικα που δεν έχει γραφεί από εσάς.
- Όλη η δουλειά σας (πηγαίος κώδικας, Makefile και README) σε ένα tar.gz file με ονομασία OnomaEponymoProject1.tar.gz. Προσοχή να υποβάλετε μόνο κώδικα, Makefile, README και όχι τα binaries.

Διαδικαστικά

- Για επιπρόσθετες ανακοινώσεις, παρακολουθείτε το forum του μαθήματος στο piazza.com. Η πλήρης διεύθυνση είναι https://piazza.com/uoa.gr/spring2020/k24/. Η παρακολούθηση του φόρουμ στο Piazza είναι υποχρεωτική.
- Το πρόγραμμά σας θα πρέπει να γραφεί σε C (ή C++). Στην περίπτωσή που χρησιμοποιήσετε C++ δεν μπορείτε να χρησιμοποιήσετε τις έτοιμες δομές της Standard Template Library (STL). Σε κάθε περίπτωση το πρόγραμμά σας θα πρέπει να τρέχει στα Linux workstations του Τμήματος.

- Το πρόγραμμά σας θα πρέπει να κάνει compile στο εκτελέσιμο (`diseaseMonitor`) και να έχει τα ίδια ονόματα για τις παραμέτρους (`-p, -h1, -h2, -b`) όπως ακριβώς περιγράφεται στην εκφώνηση.
- Ο κώδικάς σας θα πρέπει να αποτελείται από τουλάχιστον δύο (και κατά προτίμηση περισσότερα) διαφορετικά αρχεία. Η χρήση του separate compilation είναι επιτακτική και ο κώδικάς σας θα πρέπει να έχει ένα Makefile.
- Βεβαιωθείτε πως ακολουθείτε καλές πρακτικές software engineering κατά την υλοποίηση της άσκησης. Η οργάνωση, η αναγνωσιμότητα και η ύπαρξη σχολίων στον κώδικα αποτελούν κομμάτι της βαθμολογίας σας.
- Στα πρώτα μαθήματα κυκλοφορεί hard-copy λίστα (ή URL μιας ιστοσελίδας) στην οποία θα πρέπει οπωσδήποτε να δώσετε το όνομά σας και το Unix user-id σας. Με αυτό τον τρόπο μπορούμε να γνωρίζουμε ότι προτίθεστε να υποβάλετε την παρούσα άσκηση και να προβούμε στις κατάλληλες ενέργειες για την τελική υποβολή της άσκησης.

Άλλες σημαντικές παρατηρήσεις

- Οι εργασίες είναι ατομικές.
- Όποιος υποβάλει / παρουσιάσει κώδικα που δεν έχει γραφτεί από την ίδια/τον ίδιο μηδενίζεται στο μάθημα.
- Αν και αναμένεται να συζητήσετε με φίλους και συνεργάτες το πώς θα επιχειρήσετε να δώσετε λύση στο πρόβλημα, αντιγραφή κώδικα (οποιασδήποτε μορφής) είναι κάτι που δεν επιτρέπεται. Οποιοσδήποτε βρεθεί αναμεμειγμένος σε αντιγραφή κώδικα απλά παίρνει μηδέν στο μάθημα. Αυτό ισχύει για όσους εμπλέκονται ανεξάρτητα από το ποιος έδωσε/πήρε κλπ.
- Οι ασκήσεις προγραμματισμού μπορούν να δοθούν με καθυστέρηση το πολύ 3 ημερών και με ποινή 5% για κάθε μέρα αργοπορίας. Πέραν των 3 αυτών ημερών, δεν μπορούν να κατατεθούν ασκήσεις.