Εισαγωγή

Στην εργασία αυτή θα υλοποιήσετε ένα πρόγραμμα που θα δέχεται, θα επεξεργάζεται, θα καταγράφει και θα απαντάει ερωτήματα για εμβολιασμούς πολιτών. Συγκεκριμένα, θα υλοποιήσετε ένα σύνολο δομών (bloom filters, linked lists, skip lists) που επιτρέπουν την εισαγωγή και επερωτήσεις μεγάλου όγκου εγγραφών τύπου `citizenRecord`. Αν και τα δεδομένα της άσκησης θα προέρχονται από αρχεία, τελικά όλες οι εγγραφές θα αποθηκεύονται μόνο στην κύρια μνήμη.

Bloom Filter - Περιγραφή Δομής

Ένα bloom filter (βλ. https://en.wikipedia.org/wiki/Bloom_filter) είναι μία συμπαγής δομή η οποία χρησιμοποιείται για τον έλεγχο της ύπαρξης ενός στοιχείου σε ένα σύνολο. Αναλυτικότερα, χρησιμοποιώντας το bloom filter για ένα δεδομένο στοιχείο, μπορούμε να εξάγουμε ένα από τα εξής συμπεράσματα. Είτε ότι (I) το στοιχείο **σίγουρα δεν ανήκει** στο σύνολο, ή ότι (ii) το στοιχείο είναι **πιθανόν να ανήκει** στο σύνολο, αλλά δεν γνωρίζουμε με σιγουριά.

Δηλαδή το bloom filter μπορεί, σε ορισμένες περιπτώσεις, να απαντήσει θετικά ότι ένα στοιχείο ανήκει στο σύνολο, ακόμα κι όταν δεν ανήκει (false positive). Αντίθετα, η αρνητική απάντηση είναι πάντα έγκυρη, δηλαδή όταν το bloom filter υποδεικνύει πως ένα στοιχείο δεν ανήκει στο σύνολο, το στοιχείο αυτό σίγουρα δεν ανήκει. Με αυτόν τον τρόπο, το bloom filter κάνει μία υπερεκτίμηση των στοιχείων του συνόλου. Η πιθανότητα λάθους (σε περίπτωση θετικής απάντησης) αυξάνεται όσο προστίθενται στοιχεία στο σύνολο.

Ένα bloom filter αναπαρίσταται ως ένας πίνακας από bits *M* θέσεων (αρχικοποιημένος με 0), και συνοδεύεται από *K* hash functions. Ακολουθούν οι βασικές πράξεις που υποστηρίζει και το πως υλοποιούνται:

1. Για να εισάγουμε ένα στοιχείο *s* στο σύνολο, εφαρμόζουμε κάθε μία από αυτές τις *K* hash functions στο εν λόγω στοιχείο. Αυτές θα επιστρέψουν Κ θέσεις στον πίνακα (που κάποιες πιθανόν να συμπίπτουν). Έπειτα, σε κάθε μία από αυτές τις θέσεις θέτουμε το αντίστοιχο bit σε 1.
2. Για να ελέγξουμε αν ένα στοιχείο *s* ανήκει στο σύνολο, εφαρμόζουμε κάθε μία από αυτές τις *K* hash functions στο εν λόγω στοιχείο. Αν σε κάθε μία από τις *K* θέσεις που μας επιστρέφουν υπάρχουν 1, τότε το στοιχείο *s* (πιθανόν) να ανήκει στο σύνολο. Διαφορετικά (αν υπάρχει έστω κι ένα 0), σίγουρα το στοιχείο δεν ανήκει στο σύνολο.

Παράδειγμα Έστω M=16, K=3, και $hash_1()$, $hash_2()$, $hash_3()$ οι τρεις hash functions που μας έχουν δοθεί. Ο πίνακας των bits αρχικά θα είναι: 00 00 00 00 00 00 00 00.

Έστω ότι εισάγουμε διαδοχικά τα στοιχεία s_1, s_2, s_3, για τα οποία έχουμε:

$hash_1(s_1) \bmod 16 = 3$	$hash_1(s_2) \bmod 16 = 10$	$hash_1(s_3) \bmod 16 = 2$
$hash_2(s_1) \bmod 16 = 7$	$hash_2(s_2) \bmod 16 = 1$	$hash_2(s_3) \bmod 16 = 4$
$hash_3(s_1) \bmod 16 = 11$	$hash3(s_2) \bmod 16 = 7$	$hash_3(s_3) \bmod 16 = 2$

Bit array==> 00 01 00 01 00 01 00 00 ==>01 01 00 01 00 11 00 00 ==>01 11 10 01 00 11 00 00

Έστω ότι έπειτα, ελέγχουμε την ύπαρξη των στοιχείων s_4, s_5, s_3, για τα οποία έχουμε:

$hash_1(s_4) \bmod 16 = 5$	$hash_1(s_5) \bmod 16 = 2$	$hash_1(s_3) \bmod 16 = 2$
$hash_2(s_4) \bmod 16 = 6$	$hash_2(s_5) \bmod 16 = 10$	$hash_2(s_3) \bmod 16 = 4$
$hash_3(s_4) \bmod 16 = 1$	$hash_3(s_5) \bmod 16 = 11$	$hash_3(s_3) \bmod 16 = 2$

Bit array==> 01 11 10 01 00 11 00 00 ==>01 11 10 01 00 11 00 00 ==>01 11 10 01 00 11 00 00

Answer: no Answer: yes Answer: yes

Στη περίπτωση του s_5, έχουμε ένα false positive αφού το στοιχείο φαίνεται πως ανήκει στο σύνολο, ενώ στην πραγματικότητα δεν ανήκει. Οι υπόλοιπες δύο απαντήσεις είναι έγκυρες.

Skip List - Περιγραφή Δομής

Μια skip list είναι μια probabilistic δομή δεδομένων που επιτρέπει `O(log n)` πολυπλοκότητα αναζήτησης καθώς και `O(log n)` πολυπλοκότητα εισαγωγής εντός μιας ταξινομημένης λίστας `n` στοιχείων. Έτσι μπορεί να αποκτήσει τις καλύτερες δυνατότητες ενός ταξινομημένου πίνακα (για αναζήτηση) διατηρώντας παράλληλα μια δομή τύπου συνδεδεμένης λίστας που επιτρέπει την εισαγωγή, η οποία δεν είναι δυνατή σε έναν πίνακα. Η γρήγορη αναζήτηση καθίσταται δυνατή διατηρώντας μια συνδεδεμένη ιεραρχία λιστών, με κάθε διαδοχική λίστα να παραλείπει λιγότερα στοιχεία από την προηγούμενη. Μπορείτε να μάθετε περισσότερες πληροφορίες για αυτήν την δομή δεδομένων στο https://en.wikipedia.org/wiki/Skip_list. Επίσης υπάρχουν και πολλά video-tutorials στο YouTube όπως αυτό στο https://www.youtube.com/watch?v=UGaOXaXAM5M.

A) Η εφαρμογη (75%)

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

```
./vaccineMonitor -c citizenRecordsFile -b bloomSize
```

όπου:

- Η παράμετρος `bloomSize` καθορίζει το μέγεθος του bloom filter σε *bytes*. Ενδεικτικό μέγεθος του bloom filter για τα δεδομένα της άσκησης θα είναι της τάξης των 100Kbytes.
- Το `citizenRecordsFile` (ή κάποιο άλλο όνομα αρχείου) είναι ένα αρχείο που περιέχει μια σειρά από εγγραφές πολιτών προς επεξεργασία. Κάθε γραμμή του αρχείου αυτού περιγράφει την κατάσταση εμβολιασμού ενός πολίτη για κάποια συγκεκριμένη ίωση. Για παράδειγμα αν τα περιεχόμενα του αρχείου είναι:

```
889 John Papadopoulos Greece 52 COVID-19 YES 27-12-2020
889 John Papadopoulos Greece 52 H1N1 NO
776 Maria Tortellini Italy 36 SARS-1 NO
125 Jon Dupont USA 76 H1N1 YES 30-10-2020
```

σημαίνει πως έχουμε τέσσερις εγγραφές που περιγράφουν τρεις πολίτες σε τρεις διαφορετικές χώρες (Ελλάδα, Ιταλία, ΗΠΑ). `YES` υποδεικνύει πως ο πολίτης έχει εμβολιαστεί (στην ημερομηνία που ακολουθεί) ενώ το `NO` το αντίθετο. Συγκεκριμένα, μια εγγραφή είναι μια γραμμή ASCII κειμένου που αποτελείται από τα εξής στοιχεία με αυτή τη σειρά:

- `citizenID`: μια συμβολοσειρά (μπορεί να έχει και ένα μόνο ψηφίο) που καθορίζει την κάθε τέτοια εγγραφή.
- `firstName`: μια συμβολοσειρά που αποτελείται από γράμματα χωρίς κενά.
- `lastName`: μια συμβολοσειρά που αποτελείται από γράμματα χωρίς κενά.
- `country`: μια συμβολοσειρά που αποτελείται από γράμματα χωρίς κενά.
- `age`: ένας θετικός (>0), ακέραιος <= 120.
- `virusName`: μια συμβολοσειρά που αποτελείται από γράμματα, αριθμούς, και ενδεχομένως και μια παύλα "-" αλλά χωρίς κενά.
- `YES` ή `NO`: υποδεικνύει αν ο πολίτης έχει κάνει το εμβόλιο κατά τον συγκεκριμένο ιό.

- `dateVaccinated`: ημερομηνία που εμβολιάστηκε ο πολίτης. Αν το προηγούμενο πεδίο είναι `NO` , δεν υπάρχει πεδίο `dateVaccinated` στην εγγραφή.

Ξεκινώντας, η εφαρμογή σας θα πρέπει να ανοίξει το αρχείο `citizenRecordsFile` 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 `citizenRecordsFile`, υπάρχουν δύο inconsistent εγγραφές με το ίδιο `citizenID`, θα πρέπει να αγνοήσετε την δεύτερη εγγραφή. Επίσης, αν εντοπίσετε μια εγγραφή με `NO` που έχει ημερομηνία εμβολιασμού, η εγγραφή θα πρέπει να απορρίπτεται. Τις λάθος εγγραφές τις αγνοείτε κατά την ανάγνωση του αρχείου εκτυπώνοντας το μήνυμα:

`ERROR IN RECORD theRecord`
όπου `theRecord` είναι η προβληματική εγγραφή.

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

- `/vaccineStatusBloom citizenID virusName`

Η εφαρμογή θα ελέγχει το bloom filter που σχετίζεται με `virusName` και θα τυπώνει μήνυμα για το αν ο πολίτης με αριθμό ταυτότητας `citizenID` έχει κάνει το εμβόλιο κατά του `virusName`. (Δείτε παρακάτω αναλυτικά τις δομές που θα πρέπει να κρατάτε).
Output format:
```
NOT VACCINATED  OR
MAYBE
```

- `/vaccineStatus citizenID virusName`

Η εφαρμογή θα ελέγχει τη skip list που σχετίζεται με `virusName` και θα τυπώνει μήνυμα για το αν ο πολίτης με αριθμό ταυτότητας `citizenID` έχει κάνει το εμβόλιο κατά του `virusName`.
Output format:
```
NOT VACCINATED  OR
VACCINATED ON 27-12-2020
```

- `/vaccineStatus citizenID`

Η εφαρμογή θα ελέγχει όλες τις skip lists (μία για κάθε ίωση), θα εντοπίζει όλες τις εγγραφές του πολίτη με αριθμό ταυτότητας `citizenID`, και θα τυπώνει για κάθε ίωση αν έχει εμβολιαστεί και την ημερομηνία εμβολιασμού.
Output format: μια γραμμή για κάθε ίωση. Παράδειγμα:

```
COVID-19 YES 27-12-2020
SARS-1 NO
H1N1 YES 11-11-2020
```

- `/populationStatus [country] virusName date1 date2`

Αν δεν δοθεί `country` όρισμα, η εφαρμογή θα τυπώνει για την ασθένεια `virusName` τον αριθμό πολιτών σε κάθε χώρα που έχουν εμβολιαστεί μέσα στο διάστημα `[date1...date2]` και το ποσοστό του πληθυσμού της χώρας που έχει εμβολιαστεί. Αν δοθεί `country` όρισμα, η εφαρμογή θα τυπώνει για την ασθένεια `virusName`, τον αριθμό πολιτών που έχουν εμβολιαστεί και το ποσοστό του πληθυσμού της χώρας που έχει εμβολιαστεί μέσα στο διάστημα `[date1...date2]`. Εάν υπάρχει ο ορισμός για `date1` θα πρέπει να υπάρχει και ορισμός για `date2`, αλλιώς, θα τυπώνεται το μήνυμα λάθους `ERROR` στον χρήστη.
Output format: μια γραμμή για κάθε χώρα. Παράδειγμα όπου δεν έχει δοθεί `country` όρισμα:

```
GREECE 523415 5.02%
USA 358000000 10.8%
ISRAEL 3289103 38.0%
```

- `/popStatusByAge [country] virusName date1 date2`

Αν δεν δοθεί `country` όρισμα, η εφαρμογή θα τυπώνει για την ασθένεια `virusName` τον αριθμό εμβολιασμών ανά ηλικιακή κατηγορία σε κάθε χώρα και το ποσοστό του πληθυσμού της ηλικιακής κατηγορίας που έχει εμβολιαστεί μέσα στο διάστημα `[date1...date2]`. Αν δοθεί `country` όρισμα, η εφαρμογή θα τυπώνει για την ασθένεια `virusName`, τον αριθμό εμβολιασμών ανά ηλικιακή κατηγορία και το ποσοστό του πληθυσμού της ηλικιακής κατηγορίας που έχει εμβολιαστεί μέσα στο διάστημα `[date1...date2]` στη χώρα `country`. Εάν υπάρχει ο ορισμός για `date1` θα πρέπει να υπάρχει και ορισμός για `date2`, αλλιώς, θα τυπώνεται το μήνυμα λάθους `ERROR` στον χρήστη.
Output format: Παράδειγμα όπου δεν έχει δοθεί `country` όρισμα:

```
GREECE
0-20 0 0%
20-40 18795 0.36%
40-60 64650 1.24%
60+  439970 8.44%

ISRAEL
0-20 0 15%
20-40 18795 23%
40-60 64650 32.24%
60+  4399070 90.44%
```

- `/insertCitizenRecord citizenID firstName lastName country age virusName YES/NO [date]`

Η εφαρμογή θα εισάγει στο bloom filter και στη κατάλληλη skip list που σχετίζεται με την ίωση `virusName` μια νέα εγγραφή με τα στοιχεία της. Μόνο το `YES` συνοδεύεται από ένα `date`. Αν ο πολίτης με αριθμό ταυτότητας `citizenID` έχει ήδη εμβολιαστεί κατά του ιού `virusName` η εφαρμογή επιστρέφει:

```
ERROR: CITIZEN 889 ALREADY VACCINATED ON 27-12-2020
```

- `/vaccinateNow citizenID firstName lastName country age virusName`

Η εφαρμογή ελέγχει αν ο πολίτης με αριθμό ταυτότητας `citizenID` έχει ήδη εμβολιαστεί κατά του ιού `virusName` και αν ναι, επιστρέφει:
`ERROR: CITIZEN 889 ALREADY VACCINATED ON 27-12-2020.`

Αν δεν έχει εμβολιαστεί, η εφαρμογή εισάγει στο bloom filter και στη κατάλληλη skip list που σχετίζεται με την ίωση `virusName` την εγγραφή:
`citizenID firstName lastName country age virusName YES todays_date` όπου `todays_date` είναι η σημερινή ημερομηνία.

- `/list-nonVaccinated-Persons virusName`

Η εφαρμογή θα προσπελάζει την κατάλληλη skip list που σχετίζεται με την ίωση `virusName` και θα τυπώνει όλους τους πολίτες που δεν έχουν εμβολιαστεί κατά της `virusName`. Συγκεκριμένα, θα τυπώνει `citizenID`, `firstName`, `lastName`, `country` και `age`.
Output format: μια γραμμή για κάθε πολίτη. Παράδειγμα:

```
125 Jon Dupont USA 76
889 John Papadopoulos GREECE 52
```

- `/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:

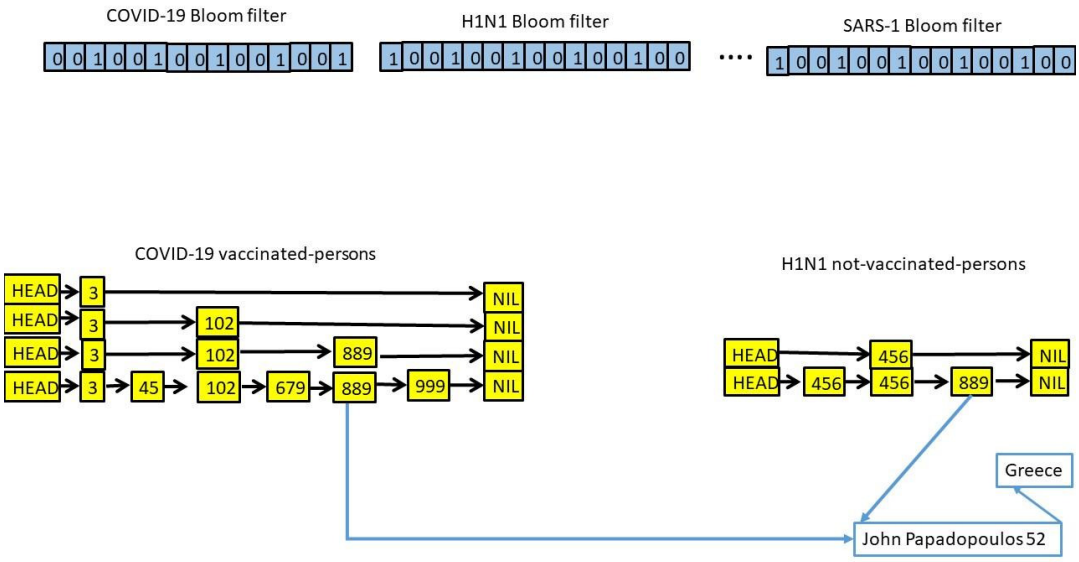
```
int countries[512];// store up to 512 countries, but really we don't know how many
```

Επίσης βεβαιωθείτε πως απελευθερώνεται η 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. Ένα bloom filter ανά ίωση που θα χρησιμοποιείται από την εφαρμογή για γρήγορο έλεγχο για το αν **δεν** είναι εμβολιασμένος ένας πολίτης.
2. Μία `vaccinated_persons` skip list ανά ίωση για να εντοπίζει η εφαρμογή πληροφορίες για πολίτες που έχουν εμβολιαστεί. Οι skip lists θα είναι ταξινομημένες με βάση το `citizenID`.
3. Μία `not_vaccinated_persons` skip list ανά ίωση για να εντοπίζει η εφαρμογή πληροφορίες για πολίτες που δεν έχουν εμβολιαστεί. Τονίζουμε πως μόνο εγγραφές με `NO` θα προστίθενται στη `not_vaccinated_persons` skip list μιας ίωσης. Στο sample input των τεσσάρων εγγραφών πιο πάνω, ο `889 John Papadopoulos` αρχικά ΔΕΝ θα προστεθεί στο `not_vaccinated_persons` skip list του `SARS-1`. Μόνο, αν στη συνέχεια, η εφαρμογη διαβάσει μια εγγραφή που γράφει ρητά `NO` για τον `889 John Papadopoulos` για `SARS-1`, θα προστεθεί στο `not_vaccinated_persons` skip list του `SARS-1`.
4. Ο στόχος σας σε αυτή την άσκηση είναι να υπάρξει όσο το δυνατόν *μικρότερη επικάλυψη στοιχείων (data duplication)*. Για παράδειγμα, οι πληροφορίες που παρέχει κάθε εγγραφή θα πρέπει να αποθηκεύονται μόνο μια φορά στην μνήμη και σε οποιαδήποτε δομή δεδομένων χρειάζεται πρόσβαση σε αυτές, η πρόσβαση θα γίνεται μέσω pointers. (δείτε Σχήμα 1 για μια πιθανή πρόταση του layout κάποιων δομών δεδομένων).
5. Όταν λαμβάνει η εφαρμογή ένα `insertCitizenRecord` ή `vaccinateNow` αίτημα, αν ο πολίτης είναι καταγεγραμμένος στο σύστημα ως μη εμβολιασμένος, θα πρέπει να ενημερωθούν οι κατάλληλες skip lists (π.χ., αφαίρεση από `not_vaccinated_persons` skip list, πρόσθεση σε `vaccinated_persons` skip list).

Βεβαιωθείτε πως για κάθε υπό-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 vaccineMonitor

B) Το script testFile.sh (25%)

Θα γράψετε ένα bash script το οποίο δημιουργεί test input file που θα χρησιμοποιήσετε για να κάνετε debugging του προγράμματός σας. Φυσικά κατά τη διάρκεια της ανάπτυξης του προγράμματός σας μπορείτε να χρησιμοποιήσετε λίγα και μικρά αρχεία για να κάνετε debug. Το script testFile.sh δουλεύει ως εξής:

```
./testFile.sh virusesFile countriesFile numLines duplicatesAllowed
```

- `virusesFile`: ένα αρχείο με ονόματα ιώσεων (ένα ανά γραμμή)
- `countriesFile`: ένα αρχείο με ονόματα χωρών (ένα ανά γραμμή)
- `numLines`: ο αριθμός γραμμών που θα έχει το file που θα φτιαχτεί
- `duplicatesAllowed`: αν είναι 0, τότε τα `citizenIDs` θα πρέπει να είναι μοναδικά, αλλιώς επιτρέπονται τα duplicate citizenIDs

Το script κάνει τα εξής:

1. Κάνει ελέγχους για νούμερα εισόδου
2. Διαβάζει τα αρχεία `virusesFile` και `countriesFile`
3. Δημιουργεί ένα αρχείο που ονομάζεται `inputFile` και τοποθετεί `numLines` γραμμές ακολουθώντας τη μορφή του input file που περιγράφεται πιο πάνω. Για το `citizenID` μπορείτε να δημιουργήσετε τυχαίους αριθμούς με τυχαίο μήκος από 1 έως 4 ψηφία Για το όνομα και επώνυμο μπορείτε να δημιουργήσετε τυχαίες συμβολοσειρές με τυχαίο μήκος από 3 έως 12 χαρακτήρες. Για ονόματα χωρών θα επιλέγει τυχαία το script κάποιο από το `countriesFile`. Για την ηλικία θα επιλέγει τυχαία έναν ακέραιο στο φάσμα [1,120]. Για virusNames, θα επιλέγει τυχαία το script κάποιο από το `virusesFile`.
4. Αν το `duplicatesAllowed` flag είναι ενεργοποιημένο, θα πρέπει να δημιουργεί και κάποια (τυχαία) records με duplicate `citizenIDs`.

Simplifying info/hints:
α) Μπορείτε να υποθέσετε πως όλοι οι μήνες έχουν 30 μέρες.
β) Ίσως σας φανεί χρήσιμη η $RANDOM Bash function. (Δείτε, π.χ. http://tldp.org/LDP/abs/html/randomvar.html)

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 (αυτό συμπεριλαμβάνει και κώδικα από το Διαδίκτυο!)**
- 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.
- Καλό θα είναι να έχετε ένα backup .tar της άσκησής σας όπως ακριβώς αυτή υποβλήθηκε σε κάποιο εύκολα προσπελάσιμο μηχάνημα (server του τμήματος, private github repository, private cloud).
- Η σωστή υποβολή ενός σωστού tar.gz που περιέχει τον κώδικα της άσκησής σας και ό,τι αρχεία χρειάζονται είναι αποκλειστικά ευθύνη σας. **Άδεια tar/tar.gz ή tar/tar.gz που έχουν λάθος και δεν γίνονται extract δεν βαθμολογούνται**.

Διαδικαστικά

- Για επιπρόσθετες ανακοινώσεις, παρακολουθείτε το forum του μαθήματος στο piazza.com. Η πλήρης διεύθυνση είναι https://piazza.com/uoa.gr/spring2021/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 (`vaccineMonitor`) και να έχει τα ίδια ονόματα για τις παραμέτρους (`-c, -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.
- θα πρέπει οπωσδήποτε να γραφτείτε στο eclass και 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. Τονίζουμε πως θα πρέπει να λάβετε τα κατάλληλα μέτρα ώστε να είναι προστατευμένος ο κώδικάς σας και να μην αποθηκεύεται κάπου που να

έχει πρόσβαση άλλος χρήστης (π.χ., η δικαιολογία «Το είχα βάλει σε ένα github repo και μάλλον μου το πήρε από εκεί», δεν είναι δεκτή.)

- 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.