Αλεξία Τοπαλίδου 1115201600286

Εντολές για το κυρίως πρόγραμμα:
- make
- ./vaccineMonitor -c testInputFile -b 100000

Εντολές για το bash script:
- ./testFile.sh virusesFile.txt countriesFile.txr 1000(ή οποιοσδήποτε αριθμός) 0(ή οποιοσδήποτε αριθμός)

Σε αυτή την εργασία έχουν υλοποιηθεί όλες οι δομές και τα ζητούμενα και το πρόγραμμα έχει χωριστεί σε αρχεία ανάλογα με τη θεματολογία για την καλύτερη οργάνωσή του. Πιο συγκεκριμένα:

hashFunctions.h και hashFunctions.c:

Αυτά τα δύο αρχεία περιέχουν τους ορισμούς και τις υλοποιήσεις των hash functions όπως ακριβώς δόθηκαν από τους καθηγητές και δεν έχει αλλαχθεί τίποτα.

fileHandling.h και fileHandling.c:

Αυτά τα δύο αρχεία περιέχουν τους ορισμούς και τις υλοποιήσεις των συναρτήσεων που χρειάζονται για το χειρισμό των αρχείων. Συγκεκριμένα:

FILE * openFile(char * f): Η συνάρτηση αυτή δέχεται το όνομα ενός αρχείου, το ανοίγει και επιστρέφει έναν δείκτη στο αρχείο αυτό.

void closeFile(FILE * f): Η συνάρτηση αυτή δέχεται το όνομα ενός αρχείου και το κλείνει.

int readFile(FILE * f): Η συνάρτηση αυτή δέχεται το όνομα ενός αρχείου το οποίο το διαβάζει γραμμή προς γραμμή και επιστρέφει τον αριθμό των γραμμών του αρχείου.

int readWords(char* Sentence, char wordArray[8][30]): Η συνάρτηση αυτή δέχεται μία πρόταση και έναν πίνακα και κάνει store τις λέξεις στον πίνακα. Επιστρέφει τον αριθμό των λέξεων.

record.h και record.c:

Αυτά τα δύο αρχεία περιέχουν τις συναρτήσεις για τη δημιουργία των records δηλαδή των δομών που έχει η εγγραφή για τους citizens. Συγκεκριμένα:

struct record: Το κάθε record είναι μια δομή με τα εξής: 1) Έναν ακέραιο που είναι το citizenID του citizen, 2) Το όνομα του citizen, 3) Το επίθετο του citizen, 4) Έναν δείκτη στην πόλη στη οποία διαμένει ο citizen, 5) έναν ακέραιο για την ηλικία του citizen.
Σχεδιαστική επιλογή της δομής του record: Η σχεδιαστική επιλογή ήταν τέτοια ώστε δεν προστέθηκε στο record το όνομα του ιού, η ημερομηνία και το yes/no διότι από τη στιγμή που θα προστεθούν τα citizenID μέσα στις δομές (bloomFilters και skipLists) είναι σαφές για το ποιο είναι το όνομα του ιού στον οποίο αναφερόμαστε και για τον αν ο citizen έχει εμβολιαστεί ή όχι για τον ιό αυτόν. Επίσης η ημερομηνία δεν υπάρχει μέσα στο record διότι όπως θα αναφερθεί παρακάτω έχει μπει μέσα στο skipList ένας δείκτης σε ημερομηνίες εφόσον ο citizen έχει εμβολιαστεί και η ημερομηνία βρίσκεται από εκεί. Έτσι πετυχαίνουμε οικονομία στο χώρο.

record* createRecord (char* citizenID, char* firstName, char* lastName, char ** country, char* age): Η συνάρτηση αυτή παίρνει ορίσματα τις μεταβλητές του record που περιγράφτηκαν από πάνω και δημιουργεί ένα record. Επιστρέφει το record και τυπώνει μήνυμα "A record has been created".

void recordFree(record* r): Η συνάρτηση αυτή απελευθερώνει τη μνήμη για το record.

countryList.h και countryList.c:

Αυτά τα δύο αρχεία περιέχουν τις συναρτήσει για τη δημιουργία και το χειρισμό λιστών που περιέχουν countries. Συγκεκριμένα:

struct countryNode: Αυτή η δομή είναι το node της λίστας. Το κάθε node περιέχει ένα country και ένα δείκτη στο επόμενο node της λίστας.

struct countryList: Αυτή η δομή περιέχει ένα δείκτη στο node που είναι η κεφαλή της λίστας.

countryNode * createCountryNode(char* country): Αυτή η συνάρτηση δημιουργεί δυναμικά ένα νέο node. Δέχεται ως όρισμα ένα country και το βάζει στο node. Προς το παρόν το node δεν έχει επόμενο οπότε το επόμενο είναι NULL.

countryList * createCountryList(): Αυτή η συνάρτηση δημιουργεί δυναμικά μια λίστα κενή που στη συνέχεια θα γεμίσει με countries.

void addCountryToList(char* country, countryList * list): Αυτή η συνάρτηση προσθέτει ένα νέο country στο τέλος της λίστας.

void displayCountries(countryList * list): Η συνάρτηση αυτή τυπώνει τις countries της λίστας.

char getNthCountry(countryList* list, int index):** Η συνάρτηση αυτή επιστρέφει έναν δείκτη στο νιοστό country της λίστας

char searchCountryList(countryList *list, char* country):** Αυτή η συνάρτηση ψάχνει τη λίστα για την ύπαρξη ενός country. Επιστρέφει δείκτη στο country ή null αν το country δεν βρεθεί.

int existCountryAtList(countryList *list, char* country): Αυτή η συνάρτηση ψάχνει τη λίστα για την ύπαρξη ενός country. Επιστρέφει 1 αν βρεθεί ή 0 αν δεν βρεθεί.

void destroyCountryList(countryList * list): Αυτή η συνάρτηση καταστρέφει τη λίστα των countries ελευθερώνοντας τη μνήμη. Τυπώνει το μήνυμα "A country list has been destroyed!".

dateVaccinatedList.h και dateVaccinatedList.c:

Αυτά τα δύο αρχεία περιέχουν τις συναρτήσει για τη δημιουργία και το χειρισμό λιστών που περιέχουν countries. Συγκεκριμένα:

struct dateVaccinatedNode: Αυτή η δομή είναι το node της λίστας. Το κάθε node περιέχει ένα date και ένα δείκτη στο επόμενο node της λίστας.

struct dateVaccinatedList: Αυτή η δομή περιέχει ένα δείκτη στο node που είναι η κεφαλή της λίστας.

dateVaccinatedNode * createdateVaccinatedNode(char* dateVaccinated): Αυτή η συνάρτηση δημιουργεί δυναμικά ένα νέο node. Δέχεται ως όρισμα ένα date και το βάζει στο node. Προς το παρόν το node δεν έχει επόμενο οπότε το επόμενο είναι NULL.

dateVaccinatedList * createdateVaccinatedList(): Αυτή η συνάρτηση δημιουργεί δυναμικά μια λίστα κενή που στη συνέχεια θα γεμίσει με dates.

void addDateVaccinatedToList(char* dateVaccinated, dateVaccinatedList * list): Αυτή η συνάρτηση προσθέτει ένα νέο date στο τέλος της λίστας.

char searchDateVaccinatedList(dateVaccinatedList *list, char* dateVaccinated):** Αυτή η συνάρτηση ψάχνει τη λίστα για την ύπαρξη ενός date. Επιστρέφει δείκτη στο date ή null αν το country δεν βρεθεί.

int existDateVaccinatedAtList(dateVaccinatedList *list, char* dateVaccinated): Αυτή η συνάρτηση ψάχνει τη λίστα για την ύπαρξη ενός date. Επιστρέφει 1 αν βρεθεί ή 0 αν δεν βρεθεί.

void destroyDateVaccinatedList(dateVaccinatedList * list): Αυτή η συνάρτηση καταστρέφει τη λίστα των dates ελευθερώνοντας τη μνήμη. Τυπώνει το μήνυμα "A dateVaccinated list has been destroyed!".

recordListFunctions.h και recordListFunctions.c:

Αυτά τα δύο αρχεία περιέχουν τις συναρτήσει για τη δημιουργία και το χειρισμό λιστών που περιέχουν records τα οποία αναφέρθηκαν παραπάνω. Συγκεκριμένα:

struct recordNode: Αυτή η δομή είναι το node της λίστας. Το κάθε node περιέχει ένα δείκτη σε record και ένα δείκτη στο επόμενο node της λίστας.

struct recordList: Αυτή η δομή περιέχει ένα δείκτη στο node που είναι η κεφαλή της λίστας.

recordNode * createRecordNode(record* rec): Αυτή η συνάρτηση δημιουργεί δυναμικά ένα νέο node. Δέχεται ως όρισμα ένα δείκτη σε record και το βάζει στο node. Προς το παρόν το node δεν έχει επόμενο οπότε το επόμενο είναι NULL.

recordList * createRecordList(): Αυτή η συνάρτηση δημιουργεί δυναμικά μια λίστα κενή που στη συνέχεια θα γεμίσει με records.

void addRecordToList(record* rec, recordList * list): Αυτή η συνάρτηση προσθέτει ένα νέο δείκτη σε record στο τέλος της λίστας.

record* searchRecordList(recordList list, char* id): Αυτή η συνάρτηση ψάχνει τη λίστα για την ύπαρξη ενός record με βάση το id του. Επιστρέφει δείκτη στο record ή null αν το record δεν βρεθεί.

int existRecordAtList(recordList *list, char* id): Αυτή η συνάρτηση ψάχνει τη λίστα για την ύπαρξη ενός record με βάση το id του. Επιστρέφει 1 αν βρεθεί ή 0 αν δεν βρεθεί.

void destroyRecordList(recordList * list): Αυτή η συνάρτηση καταστρέφει τη λίστα των records ελευθερώνοντας τη μνήμη. Τυπώνει το μήνυμα "A record list has been destroyed!".

mytok.h και mytoc.c:

Αυτά τα δύο αρχεία περιέχουν τις συναρτήσεις για το χειρισμό των γραμμών και λέξεων του αρχείου. Συγκεκριμένα:

char * getFirst(char * str): Η συνάρτηση δέχεται μια πρόταση και επιστρέφει την πρώτη λέξη. Κάθε λέξη χωρίζεται με κενό.

char * getNext(): Η συνάρτηση επιστρέφει την επόμενη λέξη. Κάθε λέξη χωρίζεται με κενό.

char * getRest(): Η συνάρτηση επιστρέφει την υπόλοιπη πρόταση.

char * getFirst1(char * str): Η συνάρτηση δέχεται μια πρόταση και επιστρέφει την πρώτη λέξη. Κάθε λέξη χωρίζεται με παύλα.

char * getNext1(): Η συνάρτηση επιστρέφει την επόμενη λέξη. Κάθε λέξη χωρίζεται με παύλα.

bitArrayFunctions.h και bitArrayFunctions.c:

Αυτά τα δύο αρχεία περιέχουν τις συναρτήσεις για το χειρισμό ενός bit array. Τα bit arrays χρησιμοποιούνται στα bloom filters. Συγκεκριμένα:

void setBit(unsigned int A[], unsigned long k): Αυτή η συνάρτηση δέχεται ως ορίσματα έναν αριθμό που αντιστοιχεί στη θέση του bit του οποίου θα θέλαμε να αλλάξουμε την τιμή του από 0 σε 1 και έναν πίνακα του οποίου θέλουμε να αλλάξουμε το K-οστό bit του.

int checkBit(unsigned int A[], unsigned long k): Αυτή η συνάρτηση ελέγχει αν το K-οστό bit ενός bit array είναι 0 ή 1. Αν είναι 0 επιστρέφει 0, αν είναι 1 επιστρέφει 1.

bloomFilterFunctions.h και bloomFilterFunctions.c:

Αυτά τα δύο αρχεία περιέχουν τις συναρτήσεις για τη δημιουργία και το χειρισμό των bloom filters. Συγκεκριμένα:

struct bloomfilter: Η δομή bloomfilter περιλαμβάνει το όνομα του bloom filter, το μέγεθος του bit array και τον bit array του bloom filter.

bloomfilter* createBloomFilter(char* name, int arraySize): Η συνάρτηση αυτή δημιουργεί δυναμικά ένα bloom filter. Παίρνει ορίσματα το όνομα του bloom filter και το μέγεθος του bit array του bloom filter σε bytes. Αρχικοποιεί τον πίνακα σε μηδενικά και επιστρέφει τον δείκτη στο bloom filter. Η συνάρτηση τυπώνει σχετικό μήνυμα όταν το bloom filter δημιουργηθεί.

int bloomFilterSearch(bloomfilter* bloomFilter, unsigned char* s): Η συνάρτηση τσεκάρει αν το s υπάρχει μέσα το bloomFilter. Αν υπάρχει επιστρέφει 1, διαφορετικά επιστρέφει 0. Η συνάρτηση hashάρει το s 16 φορές με 16 διαφορετικά hash functions και αν έστω και ένα από τα bit είναι 0 τότε το s δεν υπάρχει σίγουρα, και η συνάρτηση επιστρέφει 0. Διαφορετικά αν όλα τα bits είναι 1 τότε το s πιθανώς υπάρχει ήδη και η συνάρτηση επιστρέφει 1. Η συνάρτηση τυπώνει σχετικό μήνυμα για την ύπαρξη ή μη.

void bloomFilterInsert(bloomfilter* bloomFilter, unsigned char* s): Η συνάρτηση αυτή εισάγει το s στο bloom filter. Πρώτα κάνει έλεγχο για το αν το s υπάρχει ήδη χρησιμοποιώντας την προηγούμενη συνάρτηση. Αν δεν υπάρχει hashάρει το s και αλλάζει τα αντίστοιχα bit του bit array από 0 σε 1. Η συνάρτηση τυπώνει σχετικό μήνυμα αν έγινε η προσθήκη.

void bloomFilterFree(bloomfilter* bloom): Αυτή η συνάρτηση ελευθερώνει τη μνήμη καταστρέφοντας το bloom filter. Η συνάρτηση τυπώνει σχετικό μήνυμα.

bloomFilterListFunctions.h και bloomFilterListFunctions.c:

Αυτά τα δύο αρχεία περιέχουν τις συναρτήσει για τη δημιουργία και τον χειρισμό λιστών από bloom filters. Συγκεκριμένα:

struct bloomFilterNode: Αυτή η δομή είναι το node της λίστας. Το κάθε node περιέχει ένα δείκτη σε bloomfilter και ένα δείκτη στο επόμενο node της λίστας.

struct bloomFilterList: Αυτή η δομή περιέχει ένα δείκτη στο node που είναι η κεφαλή της λίστας.

bloomFilterNode * createBloomFilterNode(bloomfilter* bf): Αυτή η συνάρτηση δημιουργεί δυναμικά ένα νέο node. Δέχεται ως όρισμα ένα δείκτη σε bloomfilter και το βάζει στο node. Προς το παρόν το node δεν έχει επόμενο οπότε το επόμενο είναι NULL.

bloomFilterList * createBloomFilterList(): Αυτή η συνάρτηση δημιουργεί δυναμικά μια λίστα κενή που στη συνέχεια θα γεμίσει με bloom filters. Τυπώνει σχετικό μήνυμα όταν η λίστα δημιουργηθεί.

void addBloomFilterToList(bloomfilter* bf, bloomFilterList * list): Αυτή η συνάρτηση προσθέτει ένα νέο δείκτη σε bloom filter στο τέλος της λίστας. Τυπώνει σχετικό μήνυμα όταν το bloom filter προστεθεί.

bloomfilter* searchBloomFilterList(bloomFilterList *list, char* name): Αυτή η συνάρτηση ψάχνει τη λίστα για την ύπαρξη ενός bloom filter με βάση το όνομα. Επιστρέφει δείκτη στο bloom filter ή null αν το bloom filter δεν βρεθεί.

int existBloomFilterAtList(bloomFilterList *list, char* name): Αυτή η συνάρτηση ψάχνει τη λίστα για την ύπαρξη ενός bloom filter με βάση το όνομα. Επιστρέφει 1 αν το bloom filter βρεθεί και 0 αν το bloom filter δεν βρεθεί.

void destroyBloomFilterList(bloomFilterList * list): Αυτή η συνάρτηση καταστρέφει τη λίστα των bloom filters ελευθερώνοντας τη μνήμη. Τυπώνει το μήνυμα "A bloom filter list has been destroyed!".

skipList.h και skipList.c:

Αυτά τα δύο αρχεία περιέχουν τις συναρτήσεις για τη δημιουργία και το χειρισμό των skip lists. Συγκεκριμένα:

struct skiplistNode: Η δομή αυτή περιέχει έναν δείκτη σε ημερομηνία (η ημερομηνία εμβολιασμού), ένα key με βάση το οποίο ψάχνουμε τη skip list και είναι ο ακέραιος αριθμός που αντιστοιχεί στο citizenID, έναν πίνακα δεικτών σε skiplistNode's και έναν δείκτη σε record.

struct skiplist: Η δομή αυτή περιέχει το όνομα της skip list, έναν ακέραιο που δηλώνει τα επίπεδα της skip list και έναν δείκτη στην κεφαλή της skip list.

skiplistNode* createNode(int level, int key, char date, record* rec):** Η συνάρτηση αυτή δημιουργεί δυναμικά ένα node. Τα ορίσματα της συνάρτησης είναι το id του node, ο δείκτης σε ημερομηνία, ένας ακέραιος που αντιστοιχεί στο level μέχρι το οποίο θα ανέβει το node και έναν δείκτη σε record.

skiplist* createSkipList(char* listName): Η συνάρτηση αυτή δημιουργεί δυναμικά μία κενή skip list με όνομα listName. Η συνάρτηση τυπώνει σχετικό μήνυμα μετά τη δημιουργία.

int randomLevel(): Η συνάρτηση αυτή δημιουργεί με τυχαίο τρόπο έναν ακέραιο ο οποίος θα δοθεί στο κάθε node και θα αντιστοιχεί στο επίπεδο μέχρι το οποίο θα φτάσει το node.

int search(skiplist* l, int key): Η συνάρτηση αυτή ψάχνει μέσα στη skip list για ένα συγκεκριμένο id. Αν το βρει επιστρέφει 1, διαφορετικά επιστρέφει 0. Η αναζήτηση ξεκινάει από το ανώτερο επίπεδο.

record* searchRecord(skiplist* l, int key): Η συνάρτηση αυτή ψάχνει μέσα στη skip list για ένα συγκεκριμένο id. Αν το βρει επιστρέφει δείκτη σε record με αυτό το id, διαφορετικά επιστρέφει null. Η αναζήτηση ξεκινάει από το ανώτερο επίπεδο.

char* searchDate(skiplist* l, int key): Η συνάρτηση αυτή ψάχνει μέσα στη skip list για ένα συγκεκριμένο id. Αν το βρει επιστρέφει την ημερομηνία εμβολιασμού, διαφορετικά επιστρέφει null. Η αναζήτηση ξεκινάει από το ανώτερο επίπεδο.

void skiplistInsert(skiplist* l, int key, char date, record* rec):** Η συνάρτηση αυτή εισάγει ένα νέο id στη skip list. Ξεκινάει από το ανώτερο επίπεδο συγκρίνοντας τα id. Αν το id του νέου node είναι μικρότερο από το id του node στο οποίο φτάσαμε τότε γυρνάμε στο προηγούμενο node και πάμε στο από κάτω επίπεδο και συνεχίζουμε το ψάξιμο προχωρώντας μπροστά και κατεβαίνοντας μέχρι να φτάσουμε στο κατώτερο επίπεδο. Το νέο node θα έχει ύψος που παράγεται από τη συνάρτηση που αναφέρθηκε παραπάνω. Με τη βοήθεια του πίνακα

forwards συνδέουμε το νέο node με τα σωστά nodes καθώς αυτό τοποθετείται και στα πιο πάνω επίπεδα. Αν το νέο node είναι ψηλότερο από το ανώτερο επίπεδο της skip list τότε ενημερώνουμε το νέο ύψος της skip list. Η συνάρτηση τυπώνει σχετικό μήνυμα μετά την εισαγωγή.

void skiplistDelete(skiplist* l, int key): Η συνάρτηση αυτή διαγράφει ένα node με συγκεκριμένο id από τη skip list. Ξεκινάει από το ανώτερο επίπεδο και ψάχνει το node. Στη συνέχεια με τη βοήθειά του πίνακα forwards διαγράφουμε το node και από τα ανώτερα επίπεδα. Αν χρειάζεται ενημερώνουμε το νέο ύψος της skip list. Μετά τη διαγραφή, η συνάρτηση τυπώνει σχετικό μήνυμα.

int display(skiplist* l): Η συνάρτηση αυτή τυπώνει όλα τα records που έχει η skiplist l. Αν η λίστα είναι κενή επιστρέφει 0.

int citizenNumber(skiplist* l, char* country): Η συνάρτηση αυτή επιστρέφει τον αριθμό των πολιτών που βρίσκονται στην skiplist l και κατάγονται από την country.

int citizenRangeDate(skiplist* l, char* country, int day1i, int month1i, int year1i, int day2i, int month2i, int year2i): Η συνάρτηση αυτή επιστρέφει τον αριθμό των πολιτών που βρίσκονται στην skiplist l και κατάγονται από την country και εμβολιάστηκαν εντός συγκεκριμένου χρονικού διαστήματος.

int citizenRangeDate0_20(skiplist* l, char* country, int day1i, int month1i, int year1i, int day2i, int month2i, int year2i): Η συνάρτηση αυτή επιστρέφει τον αριθμό των πολιτών ηλικίας 0-20 που βρίσκονται στην skiplist l και κατάγονται από την country και εμβολιάστηκαν εντός συγκεκριμένου χρονικού διαστήματος.

int citizenRangeDate20_40(skiplist* l, char* country, int day1i, int month1i, int year1i, int day2i, int month2i, int year2i): Η συνάρτηση αυτή επιστρέφει τον αριθμό των πολιτών ηλικίας 20-40 που βρίσκονται στην skiplist l και κατάγονται από την country και εμβολιάστηκαν εντός συγκεκριμένου χρονικού διαστήματος.

int citizenRangeDate40_60(skiplist* l, char* country, int day1i, int month1i, int year1i, int day2i, int month2i, int year2i): Η συνάρτηση αυτή επιστρέφει τον αριθμό των πολιτών ηλικίας 40-60 που βρίσκονται στην skiplist l και κατάγονται από την country και εμβολιάστηκαν εντός συγκεκριμένου χρονικού διαστήματος.

int citizenRangeDate60(skiplist* l, char* country, int day1i, int month1i, int year1i, int day2i, int month2i, int year2i): Η συνάρτηση αυτή επιστρέφει τον αριθμό των πολιτών ηλικίας 60+ που βρίσκονται στην skiplist l και κατάγονται από την country και εμβολιάστηκαν εντός συγκεκριμένου χρονικού διαστήματος.

void skiplistFreeNode(skiplistNode* node): Η συνάρτηση αυτή ελευθερώνει τη μνήμη ενός node.

void skiplistFree(skiplist* l): Η συνάρτηση αυτή καταστρέφει μία skip list ελευθερώνοντας τη μνήμη. Τυπώνει σχετικό μήνυμα.

skipListListFunctions.h και skipListListFunctions.c:

Αυτά τα δύο αρχεία περιέχουν τις συναρτήσει για τη δημιουργία και τον χειρισμό λιστών από skip lists. Συγκεκριμένα:

struct skipListNode: Αυτή η δομή είναι το node της λίστας. Το κάθε node περιέχει ένα δείκτη σε skip list και ένα δείκτη στο επόμενο node της λίστας.

struct skipListList: Αυτή η δομή περιέχει ένα δείκτη στο node που είναι η κεφαλή της λίστας.

skipListNode * createSkipListNode(skiplist* sl): Αυτή η συνάρτηση δημιουργεί δυναμικά ένα νέο node. Δέχεται ως όρισμα ένα δείκτη σε skip list και το βάζει στο node. Προς το παρόν το node δεν έχει επόμενο οπότε το επόμενο είναι NULL.

skipListList * createSkipListList(): Αυτή η συνάρτηση δημιουργεί δυναμικά μια λίστα κενή που στη συνέχεια θα γεμίσει με skip lists. Τυπώνει σχετικό μήνυμα όταν η λίστα δημιουργηθεί.

void addSkipListToList(skiplist* sl, skipListList * list): Αυτή η συνάρτηση προσθέτει έναν νέο δείκτη σε skip list στο τέλος της λίστας. Τυπώνει σχετικό μήνυμα όταν η skip list προστεθεί.

skiplist* searchSkipListList(skipListList *list, char* name): Αυτή η συνάρτηση ψάχνει τη λίστα για την ύπαρξη μίας skip list με βάση το όνομα. Επιστρέφει δείκτη στη skip list ή null αν η skip list δεν βρεθεί.

int existSkipListAtList(skipListList *list, char* name): Αυτή η συνάρτηση ψάχνει τη λίστα για την ύπαρξη μίας skip list με βάση το όνομα. Επιστρέφει 1 αν βρεθεί και 0 αν δεν βρεθεί.

void destroySkipListList(skipListList * list): Αυτή η συνάρτηση καταστρέφει τη λίστα των skip lists ελευθερώνοντας τη μνήμη. Τυπώνει το μήνυμα ""A skip list, list has been destroyed!".

userCommands.h και userCommands.c:

Αυτά τα δύο αρχεία περιέχουν τις συναρτήσεις που καλεί ο χρήστης από το πληκτρολόγιο. Συγκεκριμένα:

void vaccineStatusBloom(char* citizenID, char* virusName, bloomFilterList* bl): Αυτή η συνάρτηση ελέγχει τη λίστα των bloomfilters για την ύπαρξη bloomfilter με όνομα virusName. Αν δεν βρει τυπώνει μήνυμα “There is no bloomfilter with name”. Αν το βρει αναζητά το bloomfilter για το citizenID. Αν το βρει τυπώνει “MAYBE” αν όχι τυπώνει “NOT VACCINATED”.

void vaccineStatus1(char* citizenID, char* virusName, skipListList* sll): Αυτή η συνάρτηση κάνει ότι και η προηγούμενη μόνο που τώρα ψάχνει τη skipList. Αν δεν βρει το citizenID τυπώνει μήνυμα “NOT VACCINATED” αν το βρει “VACCINATED ON ημερομηνία”.

void vaccineStatus2(char* citizenID, skipListList* vaccinatedList, skipListList* notVaccinatedList): Αυτή η συνάρτηση κάνει ότι η προηγούμενη μόνο που ψάχνει όλες τις skiplists.

void listnonVaccinatedPeople(char* virusName, skipListList* notVaccinatedList): Η συνάρτηση αυτή ψάχνει τη λίστα των skiplists των μη εμβολιασμένων. Αν εντοπίσει τη skiplist με όνομα virusName τότε τυπώνει όλα τα records των μη εμβολιασμένων πολιτών για αυτόν τον ιό. Αν η skiplist δεν βρεθεί τότε τυπώνει το μήνυμα “There is no skip list with name όνομα”. Αν η skiplist υπάρχει αλλά είναι κανή τότε τυπώνει “The skip list όνομα hasn't not vaccinated citizens”.

void populationStatus1(char* country, char* virusName, char* d1, char* d2, skipListList* notVaccinatedList, skipListList* vaccinatedList): Η συνάρτηση αυτή τυπώνει τον αριθμό και το ποσοστό του πληθυσμού που κατάγονται από την country και έχουν εμβολιαστεί για τον ιό virusName το διάστημα μεταξύ date1 και date2. Για τον υπολογισμό του ποσοστού ο παρανομαστής είναι το άθροισμα των πολιτών που κατάγονται από την country και είναι εμβολιασμένοι μαζί με αυτούς που κατάγονται από την country και δεν είναι εμβολιασμένοι.

void populationStatus2(char* virusName, char* date1, char* date2, skipListList* notVaccinatedList, skipListList* vaccinatedList, countryList* listOfCountries): Η συνάρτηση αυτή κάνει ότι η προηγούμενη αλλά για όλες τις χώρες που υπάρχουν στη λίστα των χωρών.

void popStatusByAge1(char* country, char* virusName, char* d1, char* d2, skipListList* notVaccinatedList, skipListList* vaccinatedList): Η συνάρτηση αυτή κάνει ότι η προ-προηγούμενη αλλά τυπώνει ανά ηλικιακή ομάδα τον αριθμό και ποσοστό του πληθυσμού. Ο παρανομαστής για το ποσοστό είναι ο ίδιος.

void popStatusByAge2(char* virusName, char* date1, char* date2, skipListList* notVaccinatedList, skipListList* vaccinatedList, countryList* listOfCountries): Η συνάρτηση αυτή κάνει ότι η προ-προηγούμενη αλλά τυπώνει ανά ηλικιακή ομάδα τον αριθμό και ποσοστό του πληθυσμού.

void insertCitizenRecord1(char* citizenID, char* firstName, char* lastName, char* country, char* age, char* virusName, char* yes_no, skipListList* notVaccinatedList, skipListList* vaccinatedList, bloomFilterList* bfl, recordList* recList, countryList* cl, int bloomSize): Η συνάρτηση ελέγχει αν ένας πολίτης έχει εμβολιαστεί και τυπώνει "ERROR: CITIZEN citizenID ALREADY VACCINATED ON ημερομηνία" σε κάθε άλλη περίπτωση τυπώνει "CITIZEN citizenID ADDED". Αν δεν έχει εμβολιαστεί τότε φτιάχνει νέες δομές εφόσον κάτι το συναντάμε για πρώτη φορά. Αν ο citizen υπήρχε στους μη εμβολιασμένους τότε αφαιρείται από τις αντίστοιχες δομές καθώς τώρα θα ανήκει στους εμβολιασμένους.

void insertCitizenRecord2(char* citizenID, char* firstName, char* lastName, char* coutry, char* age, char* virusName, char* yes_no, char* date, skipListList* vaccinatedList, skipListList* notVaccinatedList, bloomFilterList* bfl, recordList* recList, countryList* cl, dateVaccinatedList* dateList, int bloomSize): Η συνάρτηση ελέγχει αν ένας πολίτης έχει εμβολιαστεί και τυπώνει "ERROR: CITIZEN citizenID ALREADY VACCINATED ON ημερομηνία" σε κάθε άλλη περίπτωση τυπώνει "CITIZEN citizenID ADDED". Αν δεν έχει εμβολιαστεί τότε φτιάχνει νέες δομές εφόσον κάτι το συναντάμε για πρώτη φορά. Αν ο citizen υπήρχε στους μη εμβολιασμένους ήδη τότε τυπώνεται το μήνυμα "ERROR IN RECORD citizenID".

void vaccinateNow(char* citizenID, char* firstName, char* lastName, char* country, char* age, char* virusName, skipListList* vaccinatedList, skipListList* notVaccinatedList, bloomFilterList* bfl, recordList* recList, countryList* cl, dateVaccinatedList* dateList, int bloomSize): Η συνάρτηση ελέγχει αν ένας πολίτης έχει εμβολιαστεί και τυπώνει "ERROR: CITIZEN citizenID ALREADY VACCINATED ON ημερομηνία" σε κάθε άλλη περίπτωση τυπώνει "CITIZEN citizenID ADDED". Αν δεν έχει εμβολιαστεί τότε φτιάχνει νέες δομές εφόσον κάτι το συναντάμε για πρώτη φορά. Αν ο citizen υπήρχε στους μη εμβολιασμένους τότε αφαιρείται από τις αντίστοιχες δομές καθώς τώρα θα ανήκει στους εμβολιασμένους.

vaccineMonitor.c

Αυτό το αρχείο περιέχει τη main. Αρχικά γίνεται έλεγχος ορισμάτων. Στη συνέχεια διαβάζεται το αρχείο που δημιουργήθηκε από το bash script γραμμή γραμμή και δημιουργούνται και γεμίζουν όλες οι δομές. Καθώς διαβάζεται το αρχείο όταν συναντάμε ένα id το οποίο το έχουμε ξανά συναντήσει το απορρίπτουμε μονάχα εφόσον η εγγραφή αναφέρεται στον ίδιο ιό με την πρώτη φορά. Στη συνέχεια ζητείται από το χρήστη να δώσει εντολές. Όταν ο χρήστης πληκτρολογήσει /exit τότε απελευθερώνεται όλη η μνήμη και το πρόγραμμα τερματίζει.

Bash Script

Στην αρχή του bash script γίνονται όλοι οι απαραίτητοι έλεγχοι αν ο χρήστης:

- έδωσε σωστό αριθμό ορισμάτων από τη γραμμή εντολών
- έδωσε στο πρώτο όρισμα το όνομα ενός αρχείου που υπάρχει και είναι readable
- έδωσε στο δεύτερο όρισμα το όνομα ενός αρχείου που υπάρχει και είναι readable
- έδωσε στο τρίτο όρισμα αριθμό και θετικό
- έδωσε στο τέταρτο όρισμα αριθμό και μη αρνητικό

Στη συνέχεια το πρόγραμμα τυπώνει τα περιεχόμενα των αρχείων με τις χώρες και τους ιούς.
Τέλος δημιουργείται το αρχείο στο οποίο εγγράφονται τα records. Αν τα duplicates επιτρέπονται τότε αποφασίζεται με τυχαίο τρόπο αν μια εγγραφή θα επαναληφθεί αλλά ίσως με διαφορετικό ιό, YES/NO και ημερομηνία. Εγγραφές που έχουν NO δεν έχουν ημερομηνία.
Αν ξανατρέξουμε το bash script τότε κάθε φορά στην αρχή διαγράφεται το προηγούμενο αρχείο (αν υπάρχει) στο οποίο γράφτηκαν τα records και δημιουργείται ένα ολοκαίνουριο.