ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
Τμήμα Πληροφορικής και Τηλεπικοινωνιών
1η Εργασία - Τμήμα: Αρτίων Αριθμών Μητρώου
K22: Λειτουργικά Συστήματα – Χειμερινό Εξάμηνο '19
Ημερομηνία Ανακοίνωσης: Τρίτη 1 Οκτωβρίου 2019
Ημερομηνία Υποβολής: Πέμπτη 24 Οκτωβρίου 2019 Ώρα 23:59

Εισαγωγή στην Εργασία:

Ο στόχος αυτής της εργασίας είναι να εξοικειωθείτε με το περιβάλλον `Linux` και σχετικά εργαλεία προγραμματισμού και ανάπτυξης λογισμικού.

Θα υλοποιήσετε ένα πρόγραμμα που ονομάζεται runelection που με την βοήθεια μιας σύνθεσης δομών επιτρέπει να διεξαχθεί μια ψηφοφορία. Η σύνθεση δομών βοηθά την έφορο της ψηφοφορίας να γνωρίζει οποιαδήποτε χρονική στιγμή ποια/ος ήδη έχει συμμετάσχει, ποιος/-α δεν μπορεί να ψηφίσει, καθώς επίσης και να παράγει στατιστικά σε σχέση με τους ταχυδρομικούς κώδικες των ψηφοφόρων.

Η σύνθεση δομών που θα πρέπει να δημιουργήσετε αποτελείται από

α) ένα `Bloom Filter` [1, 2] το οποίο γρήγορα αποφασίζει αν κάποια *δεν βρίσκεται* στον εκλογικό κατάλογο.
β) ένα `Red-Black Tree (RBT)` το οποίο παρέχει παρέχει γρήγορη (λογαριθμική) προσπέλαση στην έγγραφή του εκλογέα και ανάλογα θέτει μια σχετική σημαία αν ο εν-λογω πολίτης έχει ολοκληρώσει την διαδικασία,
γ) οποιαδήποτε άλλη δομή κρίνετε απαραίτητη για να δημιουργήσετε ομάδες ψηφοφόρων που προέρχονται από τον ίδιο ταχυδρομικό κώδικα.

Όταν μια ψηφοφορία ξεκινά, το πρόγραμμα σας θα πρέπει να δημιουργεί τις παραπάνω 3 δομές ώστε η έφορος να μπορεί να ελέγχει αν ένας ψηφοφόρος μπορεί να ψηφίσει και αν μπορεί να θέτει το ανάλογο `hasvoted` flag του/της στην δομή `RBT`.

Μερικές βασικές προϋποθέσεις για την παραπάνω εφαρμογή είναι οι εξής:

1. Ο κατάλογος εκλεκτόρων είναι δεδομένος και είναι διαθέσιμος την στιγμή που γίνεται η κλήση του `runelection`.
2. Η εφαρμογή, αν κάποιος ψηφίσει, μπορεί να θέτει ανάλογα τη σημαία `hasvoted` ή να αποτρέπει δεύτερη ή/και μη επιτρεπόμενη ψηφοφορία.
3. Η έφορος θα μπορεί να εισαγάγει στις δομές κατ' εξαίρεση ψηφοφόρους (που για οποιαδήποτε λόγο δεν είχαν συμπεριληφθεί στον αρχικό κατάλογο).
4. Σε οποιαδήποτε στιγμή της εκτέλεσης του `runelection` διάφορα στατιστικά μπορεί αν εκμαιευθούν από τις δομές.
5. Το πρόγραμμα σας θα πρέπει στο τέλος να τυπώνει σε ένα αρχείο εξόδου επιλογής σας στατιστικά στοιχεία για την ψηφοφορία και φυσικά να ελευθερώνει σταδιακά όλη την μνήμη που έχει δεσμεύσει.

Διαδικαστικά:

Το πρόγραμμά σας θα πρέπει να γραφτεί σε *C* (ή *C++* αν θέλετε αλλά χωρίς την χρήση STL/Templates) και να τρέχει στις μηχανές `Linux workstations` του τμήματος.

Το πρόγραμμα σας (source code) πρέπει να αποτελείται από ***τουλάχιστον δυο*** (και κατά προτίμηση πιο πολλά) διαφορετικά αρχεία. Το πρόγραμμά σας θα πρέπει *απαραιτήτως να κάνει χρήση* separate compilation .

Παρακολουθείτε την ιστοσελίδα του μαθήματος `http://www.di.uoa.gr/~ad/k22/` για επιπρόσθετες ανακοινώσεις αλλά και την ηλεκτρονική-λίστα (η-λίστα) του μαθήματος στο URL `https://piazza.com/uoa.gr/fall2019/k22/home`

- Υπεύθυνοι για την άσκηση αυτή (ερωτήσεις, αξιολόγηση, βαθμολόγηση κλπ.) είναι ο κ. Μιχάλης Κωνσταντόπουλος `mkonstan+AT-di`, ο κ. Δημήτρης Σπυρόπουλος `jimsp+AT-di`, και ο κ. Χρήστος Παπαδόπουλος `sdi1400145+AT-di`.
- Στην διάρκεια των πρώτων μαθημάτων κυκλοφορούμε hard-copy λίστα στην τάξη στην οποία θα πρέπει να δώσετε το όνομά σας και το `Linux` user-id σας.

  Με αυτό το τρόπο μπορούμε να γνωρίζουμε ποιος/-α προτίθεται να υποβάλλει την πρώτη άσκηση *ώστε να προβούμε στις κατάλληλες ενέργειες για την υποβολή της εργασίας σας*.
- Με βάση την παραπάνω λίστα, θα σας γράψουμε επίσης στο σύστημα `piazza.com` που βρίσκεται στο `https://piazza.com/uoa.gr/fall2019/k22/home`. Μόλις *αποδεχτείτε* ένα 'κωδικό' που θα σας σταλεί, μπορείτε να κάνετε ερωτήσεις και να δείτε απαντήσεις ή/και διευκρινήσεις που δίνονται σχετικά με την άσκηση.

Χαρακτηριστικά του runelection:

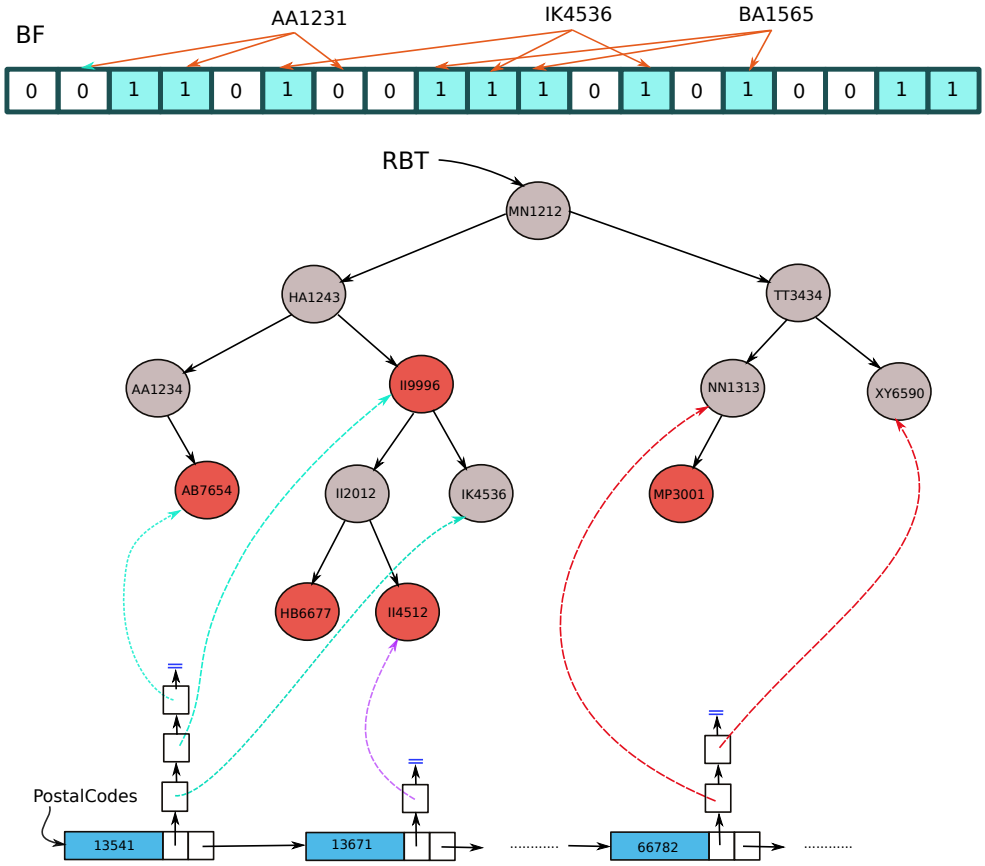
Μερικά βασικά χαρακτηριστικά που θα πρέπει να λάβετε υπ' όψη είναι:

1. Τα στοιχεία ψηφοφόρων περιλαμβάνουν: αριθμό ταυτότητας, όνομα, επίθετο, ηλικία, φύλο, και ταχυδρομικό κώδικα.
2. Το αρχείο ψηφοφόρων –`registry`– είναι μορφής ASCII, κάθε εγγραφή του περιγράφει ένα ψηφοφόρο, και το αρχείο είναι τυχαίου μεγέθους.
3. Καθώς το πρόγραμμα σας αρχίζει και δημιουργεί τις απαραίτητες δομές, ο *αριθμός ταυτότητας* (alphanumeric μορφής) λειτουργεί σαν το βασικό *κλειδί* για το `BF` και `RBT`.
4. Το μέγεθος του bit-string `BF` είναι συνάρτηση του αριθμού των εγγραφών που έρχονται από το αρχείων ψηφοφόρων και μπορεί να είναι ένας πρώτος αριθμός που είναι τουλάχιστον 3-πλασιος του αριθμού εγγραφών.
5. Για να δημιουργήσετε το `BF` μπορείτε να χρησιμοποιήσετε 3 διαφορετικές hash functions δικιά σας επιλογής.
6. Το `RBT` είναι η δομή που 'βαστά' όλες τις εγγραφές ψηφοφόρων και για κάθε μία ψηφοφόρο έχει και το επιπρόσθετο πεδίο `hasvoted` ώστε να υπάρχει ένδειξη αν ο ψηφοφόρος έχει ήδη λάβει μέρος στην διαδικασία.
7. Το πρόγραμμα σας θα πρέπει να εξασφαλίζει γρήγορη προσπέλαση σε όσες/-ους ψηφοφόρους υπάρχουν από ένα συγκεκριμένο ταχυδρομικό κωδικό ώστε να μπορείτε εύκολα να απαντήσετε συγκεκριμένα ερωτήματα για την εν λόγω γεωγραφική περιοχή.
8. Το runelection αφού εισαγάγει τα στοιχεία από το αρχείο `registry` θα μπορεί να δεχτεί από ένα `prompt` εντολές για την διεκπεραίωση της ψηφοφορίας.
9. Γενικά οποιαδήποτε μορφή υλοποίησης που βασίζεται σε *στατικούς* πίνακες *δεν είναι αποδεκτή*. Ακόμα και η χρήση δυναμικών πινάκων δεν θεωρείται η ενδεδειγμένη προσέγγισή στην άσκησή αυτή.
10. Το Σχήμα 1 δείχνει μια αντιπροσωπευτική κατάσταση της σύνθεσης δομών που θα πρέπει να δημιουργήσετε. Το `BF` ουσιαστικά ελέγχει αν κάποιος σίγουρα δεν είναι μέλος του εκλογικού καταλόγου και οι υπόλοιπες δομές διαχειρίζονται εγγραφές και την κατάσταση τους όσον αφορά στην ψηφοφορία.

Γραμμή Κλήσης της Εφαρμογής:

Η εφαρμογή μπορεί να κληθεί με τον παρακάτω αυστηρό τρόπο στην γραμμή εντολής (`tty`):
./runelection -i inputfile -o outfile -n numofupdates
όπου

- runelection είναι το εκτελέσιμο που θα δημιουργήσετε,
- inputfile είναι το αρχείο εισόδου δεδομένων. Το αρχείο είναι το registry των ψηφοφόρων, περιέχει μια σειρά από εγγραφές που η κάθε μία αποτελείται από τον Αριθμο Αστυν. Ταυτότητας, Όνομα, Επίθετο, Ηλικία,



Σχήμα 1: Παράδειγμα πιθανής οργάνωσης δομών `runelection`: Το BF ελέγχει αν ένα κλειδί δεν ανήκει στον εκλογικό κατάλογο. Αυτό ισχύει για την εγγραφή με κλειδί AA1231 που δεν υπάρχει στον κατάλογο. Για να πιστοποιηθεί ότι η εγγραφή με το κλειδί IK4536 είναι διαθέσιμη θα πρέπει να υπάρξει επιτυχής προσπέλαση της σχετικής εγγραφής στο RBT. Η δομή PostCodes θα πρέπει να συσχετίζει εγγραφές ανά ταχυδρομικό κώδικα. Η εικόνα δείχνει μια τέτοια πιθανή οργάνωση βασισμένη σε διασυνδεδεμένες λιστες που με την βοήθεια των διακοπτόμενων γραμμών (pointers) παρέχουν τις σχετικές ομάδες.

Φύλο, και Ταχυδρομικό κώδικα κατοικίας.

- outfile είναι το αρχείο εξόδου που καταγράφει μηνύματα λάθους και στο τέλος δίνει μια ολοκληρωμένη εικόνα του `registry` ειδικά αν έχουν προκύψει νέες εισαγωγές.
- numofupdates είναι μια σταθερά που δηλώνει τον αριθμό των εισαγωγών/διαγραφών που μπορεί να γίνουν στο RBT ώστε να υπάρξει αναδημιουργία του BF. Αν δεν παρέχεται μια τέτοια σταθερά, η τιμή της εξ' ορισμού είναι 5. Αν π.χ. υπάρξουν 5 νέες εισαγωγές ή διαγραφές εγγραφών στο RBT, το BF θα πρέπει να αναπαραχθεί από την αρχή λαμβάνοντας υπόψη τις «παλιές» αλλά και τις 5 «νέες» εγγραφές.

Οι παραπάνω in-line παράμετροι (και αντίστοιχες σημαίες) είναι προαιρετικές όσον αφορά την κλήση τους στην γραμμή εντολής. Ωστόσο η υλοποίησή τους είναι υποχρεωτική. Οι σημαίες μπορούν να εμφανίζονται με οποιαδήποτε σειρά.

Περιγραφή της Διεπαφής του Προγράμματος:

Αφού το πρόγραμμα κληθεί, και οι σχετικές δομές για όλους τους εγγεγραμμένους ψηφοφόρους εισαχθούν, η/ο χρήστης μπορεί να αλληλεπιδράσει με τα δεδομένα του `registry` μέσω ενός prompt. Μέσω του τελευταίου ένας αριθμός από εντολές μπορούν να κληθούν:

1. `lbf key` (lookup bloom-filter):
   έλεγξε το BF αν πιθανά εμπεριέχει το `key`. Η απάντηση μπορεί να είναι α) όχι β) πιθανόν.
2. `lrb key` (lookup red-black tree):
   έλεγξε αν το RBT εμπεριέχει εγγραφή με `key`. Αν υπάρχει η σχετική εγγραφή, παρουσίασε την.
3. `ins record` (insert record to red-black-tree):
   εισήγαγε στις δομές μια νέα εγγραφή (`record`) της οποίας όλα τα στοιχεία παρέχονται σε 1 γραμμή. Η εισαγωγή `numofupdates` εγγραφών δημιουργεί ανάγκη συνολικής αναδιοργάνωσης του BF. Στα πλαίσια αυτής της αναδιοργάνωσης παλιές και νέες εγγραφές θα πρέπει να ληφθούν υπ' όψη.
4. `find key`:
   βρες αν στην σύνθεση δομών υπάρχει ή εγγραφή με `key` και τύπωσε τη. Διαφορετικά τύπωσε σχετικό μήνυμα.
5. `delete key`:
   διέγραψε από τις δομές την εγγραφή με κλειδί `key`. Η διαγραφή δεν επηρεάζει το BF.
6. `vote key`:
   άλλαξε την κατάσταση του εκλογέα με αριθμό αστυνομικής ταυτότητας `key` όσον αφορά στο `hasvoted` πεδίο. Αν ο εκλογέας δεν υπάρχει ή αν υπάρχει και έχει ήδη ψηφίσει παρήγαγε σχετικά μηνύματα.
7. `load fileofkeys`:
   για όλα τα `keys` που υπάρχουν στο αρχείο κειμένου `fileofkeys` επιχειρούμε ψήφο (δηλ. vote για κάθε κλειδί που παρέχεται στο αρχείο `fileofkeys`).
8. `voted`:
   παρουσίασε πόσοι εκλογείς έχουν ήδη ψηφίσει.
9. `voted postcode`:
   για ένα συγκεκριμένο `postcode` παρουσίασε πόσοι εκλογείς έχουν ψηφίσει.
10. `votedperpc`:
    για κάθε `postcode` δώσε το ποσοστό την ψηφισάντων μέχρι στιγμής.
11. `exit`
    το πρόγραμμα απλά τερματίζει αφού ελευθερώσει πρώτα με συγκροτημένο τρόπο όλο το χώρο που έχει καταλάβει στην μνήμη.

Η *μορφή εξόδου* της κάθε μία από τις παραπάνω εντολές δίνεται με *λεπτομέρεια* στην σελίδα του μαθήματος.

Χαρακτηριστικά του Προγράμματος που Πρέπει να Γράψετε:

1. Δεν μπορείτε να κάνετε pre-allocate οποιοσδήποτε χώρο αφού η δομή(-ές) θα πρέπει να μπορεί(-ουν) να μεγαλώσει(-ουν) χωρίς ουσιαστικά κανέναν περιορισμό όσον αφορά στον αριθμό των εγγραφών που μπορούν να αποθηκεύσουν. Η χρήση στατικών πινάκων/δομών που δεσμεύονται στην διάρκεια της συμβολομετάφρασης του προγράμματος σας δεν είναι αποδεκτές επιλογές.
2. Για δομές που θα υιοθετήσετε, θα πρέπει να μπορείτε να εξηγήσετε (και να δικαιολογήσετε στην αναφορά σας) την πολυπλοκότητα που παρουσιάζουν.
3. Πριν να τερματίσει η εκτέλεση της εφαρμογής, το περιεχόμενο των δομών απελευθερώνεται με σταδιακό και ελεγχόμενο τρόπο.
4. Αν πιθανώς κάποια κομμάτια του κωδικά σας προέλθουν από κάποια δημόσια πηγή, θα πρέπει *να δώσετε αναφορά στη εν λόγω πηγή* είτε αυτή είναι βιβλίο, σημειώσεις, Internet URL κλπ. και να εξηγήσετε πως ακριβώς χρησιμοποιήσατε την εν λόγω αναφορά.
5. Έχετε πλήρη ελευθερία να επιλέξετε το τρόπο με τον οποίο τελικά θα υλοποιήσετε βοηθητικές δομές.

Τι πρέπει να Παραδοθεί:

1. Μια σύντομη και περιεκτική εξήγηση για τις επιλογές που έχετε κάνει στο σχεδιασμό του προγράμματος σας (2-3 σελίδες σε ASCII κειμένου είναι αρκετές).
2. Οπωσδήποτε ένα `Makefile` (που να μπορεί να χρησιμοποιηθεί για να γίνει αυτόματα το compile του προγράμματος σας). Πιο πολλές λεπτομέρειες για το (`Makefile`) και πως αυτό δημιουργείται δίνονται στην ιστοσελίδα του μαθήματος.
3. Ένα `zip/tar-file` με όλη σας την δουλειά σε έναν κατάλογο που πιθανώς να φέρει το όνομα σας και θα περιέχει όλη σας την δουλειά δηλ. `source files, header files, output files` (αν υπάρχουν) και οτιδήποτε άλλο χρειάζεται.

Άλλες Σημαντικές Παρατηρήσεις:

1. Οι εργασίες είναι **ατομικές**.
2. Το πρόγραμμα σας θα πρέπει να τρέχει στα `Linux` συστήματα του τμήματος αλλιώς **δεν μπορεί να βαθμολογηθεί**.
3. Αν και αναμένεται να συζητήσετε με φίλους και συνεργάτες το πως θα επιχειρήσετε να δώσετε λύση στο πρόβλημα, αντιγραφή κώδικα (οποιαδήποτε μορφής) είναι κάτι που **δεν επιτρέπεται** και δεν πρέπει να γίνει. Οποιοσδήποτε βρεθεί αναμεμειγμένος σε αντιγραφή κώδικά **απλά παίρνει μηδέν** στο μάθημα. Αυτό ισχύει για **όσους εμπλέκονται ανεξάρτητα** από το ποιος έδωσε/πήρε κλπ.
4. Το παραπάνω ισχύει αν διαπιστωθεί *έστω και μερική άγνοια* του κώδικα που έχετε υποβάλει ή *άπλα υπάρχει υποψία* ότι ο κώδικας είναι προϊόν συναλλαγής με τρίτο/-α άτομο/α.
5. Προγράμματα που **δεν χρησιμοποιούν** separate compilation χάνουν αυτόματα 5% του βαθμού.
6. Σε καμιά περίπτωση τα Windows *δεν είναι επιλέξιμη* πλατφόρμα για την παρουσίαση αυτής της άσκησης.

Αναφορές

[1] Wikipedia, *Bloom Filter*, *Available at*: `https://en.wikipedia.org/wiki/Bloom_filter`

[2] Andrei Broder and Michael Mitzenmacher, *Network Applications of Bloom Filters: a Survey*, Internet Mathematics, vol. 1, no. 4, 2002 *Available at*: `www.cs.rochester.edu/users/faculty/stefanko/Teaching/13CS484/BM.pdf`