ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
Τμήμα Πληροφορικής και Τηλεπικοινωνιών
1η Εργασία - Τμήμα: Περιττών Αριθμών Μητρώου
K22: Λειτουργικά Συστήματα – Χειμερινό Εξάμηνο '16
Ημερομηνία Ανακοίνωσης: Τρίτη 4 Οκτωβρίου 2016
Ημερομηνία Υποβολής: Κυριακή 23 Οκτωβρίου 2016 Ώρα 23:59

Εισαγωγή στην Εργασία:

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

Θα πρέπει να υλοποιήσετε μια δομή στην κυρίως μνήμη στην οποία να μπορείτε να εισάγετε, προσπελάσετε και να τροποποιείτε εγγραφές που έχουν να κάνουν με την πρόοδο φοιτητών του ΕΚΠΑ. Το μέγιστο κόστος εισαγωγής μίας εγγραφής πρέπει να είναι $O(n)$ όπου n είναι ο αριθμός εγγραφών που υπάρχουν ήδη αποθηκευμένες στην δομή σας. Ανάλογα με το τύπο ερώτησης που θέτουμε το κόστος προσπέλασης πρέπει να είναι είτε $O(1)$ ή $O(n)$. Για εξειδικευμένες επερωτήσεις εύρους μπορούμε να έχουμε και ένα γραμμικό κόστος $O(k)$ όπου k είναι ο αριθμός των εγγραφών ενδιαφέροντος.

Για να υπάρχει γρήγορη ($O(n)$) προσπέλαση στις εγγραφές που θα αποθηκευτούν στην δομή θα πρέπει να χρησιμοποιήσετε την δομή skip-list [1]. Ωστόσο για να επιτύχετε γρήγορες ($O(k)$) επερωτήσεις εύρους θα πρέπει να χρησιμοποιήσετε κατακερματισμό. Ο κατακερματισμός που θα πρέπει να χρησιμοποιήσετε σε αυτή την άσκηση μπορεί να είναι κάτι απλό όπως το *chained hashing*. Όσο αφορά στην skip list, παρέχουμε μια σύντομη *περιγραφή* [2].

Οι εγγραφές που θα χρησιμοποιήσετε έχουν να κάνουν με φοιτητές στο ΕΚΠΑ, την μέχρι στιγμή επίδοσή τους, τις υποχρεώσεις τους μέχρι να ολοκληρώσουν τις σπουδές τους σε ένα συγκεκριμένο πεδίο επιστήμης αλλά και την περιοχή που διαμένουν (ταχυδρομικός κώδικας). Η εφαρμογή σας πρέπει να δίνει την δυνατότητα σε υπαλλήλους του Πανεπιστημίου να μπορούν να εισάγουν/ενημερώσουν εγγραφές, να κάνουν επερωτήσεις για συγκεκριμένους φοιτητές και να εξάγουν βασικά στατιστικά στοιχεία της προόδου των φοιτητών.

Μερικές βασικές προϋποθέσεις για την παραπάνω εφαρμογή είναι οι εξής:

1. Η βασική δομή των δεδομένων –που θα πρέπει να είναι μία skip list– οργανώνεται με βάση τον αριθμό φοιτητικού μητρώου.
2. Οι εγγραφές αποτελούνται από: τον αριθμό φοιτητικού μητρώου (`studid`) που είναι μοναδικός αριθμός, επίθετο (`lastname`), όνομα (`firstname`), μέσος όρος (`gpa`), αριθμός από μαθήματα (`numofcourses`) που απομένουν για την ολοκλήρωση σπουδών, τμήμα σπουδών (`deprt`) και ταχυδρομικό κώδικα μόνιμης διαμονής (`postcode`).
3. Η παραπάνω δομή δέχεται εισαγωγές, επερωτήσεις και διαγραφές για φοιτητές, καθώς επίσης και αλλαγές στον μέσο όρο και τα μαθήματα που απομένουν για την ολοκλήρωσή των σπουδών ενός φοιτητή.
4. Η δομή γραμμικού κατακερματισμού θα χρησιμοποιείται ώστε να επιτύχουμε $O(1)$ χρόνο προσπέλασης στην βασική δομή που διαθέτει τις εγγραφές. Ωστόσο η προσπέλαση αυτή θα γίνεται με βάση το `postcode`.
5. Η εφαρμογή θα πρέπει να μπορεί εάν είναι επιθυμητό να διαβάσει δεδομένα (με λειτουργίες) στην εκκίνηση της από ένα αρχείο εισόδου.
6. Το πρόγραμμα σας θα πρέπει –όποτε αυτό απαιτείται– να ελευθερώνει με *εύλογο τρόπο* την μνήμη που έχει δεσμεύσει. Το ίδιο ισχύει για τον τερματισμό της εφαρμογής.

Μέθοδοι προσπέλασης σαν και αυτές που αναφέρονται παραπάνω είναι πολύ κοινές σε υλοποιήσεις συστημάτων ανάκτησης πληροφορίας, μηχανών αναζήτησης, και πληροφοριακών συστημάτων.

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. Η χρήση του separate compilation είναι *επιτακτική!*

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/fall2016/k22/home`

- Υπεύθυνοι για την άσκηση αυτή (ερωτήσεις, αξιολόγηση, βαθμολόγηση κλπ.) ο κ. Παναγιώτης Βλαντής `panosv+AT-di`, ο κ. Κωνσταντίνος Αποστολόπουλος `apostkonst+AT-di` και ο κ. Θέμης Μπερής `sdi1200121+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/Unix` user-id σας.

  Με αυτό το τρόπο μπορούμε να γνωρίζουμε ποιος/-α προτίθεται να υποβάλλει την πρώτη άσκηση και να προβούμε στις κατάλληλες ενέργειες για την υποβολή της άσκησης.
- Με βάση την παραπάνω λίστα, θα σας γράψουμε στο σύστημα `piazza.com` και πιο συγκεκριμένα στο `https://piazza.com/uoa.gr/fall2016/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.

Η Σύνθεση Δομής που θα Δημιουργήσετε:

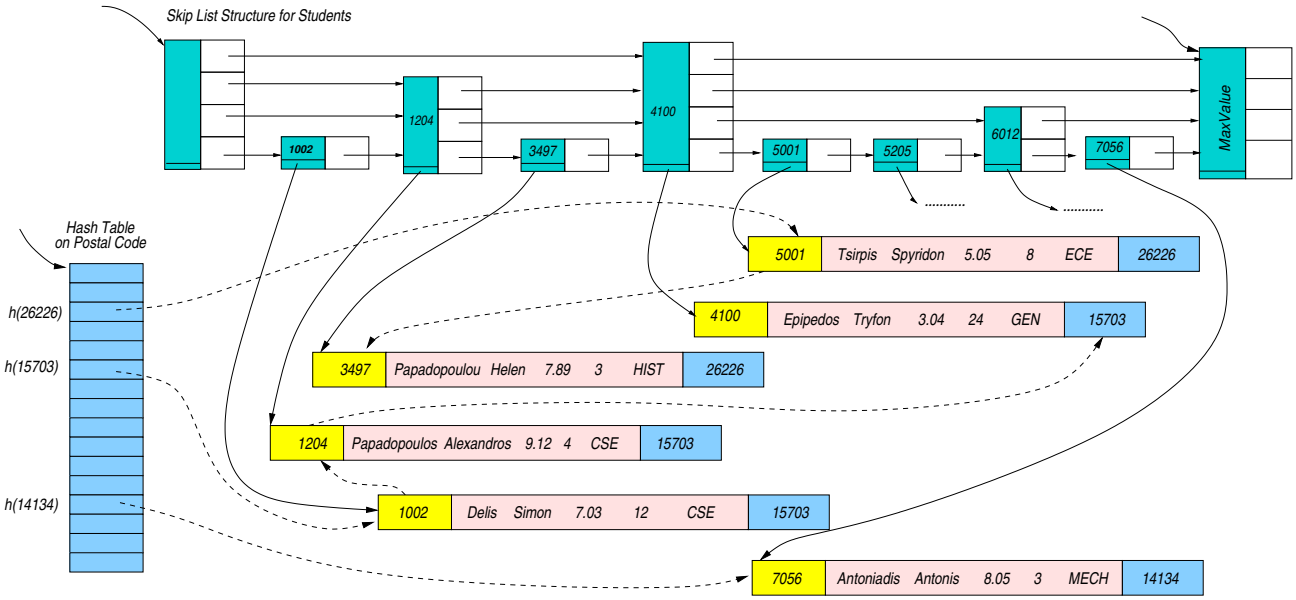
Το Σχήμα 1 δείχνει μια μερική αλλά αντιπροσωπευτική κατάσταση της σύνθετης δομής που θα δημιουργήσετε και ονομάζεται `stucs (student statistics)`.

Οι εγγραφές φοιτητών είναι οργανωμένες κατά βάση με την βοήθεια skip-list σε *αύξουσα* σειρά αριθμού μητρώου όπως δείχνει το κύριο τμήμα του Σχήματος 1. Κάθε εγγραφή έχει όλα τα στοιχεία πληροφορίας για τον φοιτητή και και περιέχει το `studid` (κλειδί), επίθετο, όνομα, μέσο όρο, αριθμό από μαθήματα, τμήμα και ταχυδρομικό κώδικα. Προφανώς αλλαγές στους μέσους όρους και το αριθμό μαθημάτων που κάποια φοιτήτρια χρειάζεται να πάρει ακόμα για να ολοκληρώσει σπουδές, μπορούν να γίνουν με την βοήθεια της παραπάνω skip-list δομής.

Η προσπέλαση στις εγγραφές επιτυγχάνεται και με την βοήθεια chained hashing που βασίζεται στην χρήση του `postcode` σαν κλειδί. Το αριστερό τμήμα του Σχήματος 1 βοηθά να γίνει κατανοητή η χρήση του κατακερματισμού στην περίπτωσή της άσκησης. Όλοι οι φοιτητές που διαμένουν στο `postcode` με νούμερο `15703` θα κάνουν 'hash' στην ίδια θέση του πίνακα. Έτσι μια υπάλληλος μπορεί να βρει άμεσα *όλες* τις φοιτήτριες που ζουν στην περιοχή `15703` ακολουθώντας τους συνδέσμους με τις διακεκομμένες γραμμές. Για παράδειγμα στο Σχήμα 1, οι φοιτητές που μένουν στην περιοχή με κωδικό `15703` είναι οι *Δελής, Παπαδόπουλος* και *Επίπεδος*. Στην πραγματικότητα δεν είναι απαραίτητο σε αυτή την λίστα να βρίσκονται ΜΟΝΟ φοιτητές που μένουν στον `15703` καθώς και άλλοι ταχυδρομικοί κώδικες μπορούν να κάνουν hash στην ίδια θέση του πίνακα κατακερματισμού.

Ο αριθμός των αναμενόμενων εισαγωγών/προσαυξήσεων εγγραφών στην δομή δεν είναι γνωστός εξ' αρχής. Οι δύο δομές του Σχήματος 1 μπορούν να μεγαλώσουν και να τροποποιηθούν κατά βούληση.

Η συγκεκριμένη μορφή που παίρνουν ο πίνακας καταμερισμού και η δομή αποθήκευσης έχουν να κάνουν με επιλογές σχεδιασμού που θα πρέπει να πάρετε και να περιγράψετε στο σύντομο αρχείο σχεδιασμού που θα υποβάλετε μαζί με τον κώδικα σας. Συνολικά ωστόσο οι δομές που τελικά θα χρησιμοποιήσετε θα πρέπει να



Σχήμα 1: Παράδειγμα οργάνωσης της δομής με skip-list να χρησιμοποιεί το studid σαν κλειδί και το κατακερματισμό να χρησιμοποιεί το postcode σαν κλειδί.

είναι *δυναμικές*. Σημειώστε ότι η χρήση στατικών πινάκων η και ακόμα δυναμικά παρεχόμενων πινάκων για την αποθήκευση των δεδομένων του Σχήματος 1 δεν είναι αποδεκτή στην εν λόγω άσκηση.

Γραμμή Κλήσης της Εφαρμογής:

Η εφαρμογή μπορεί να κληθεί με τον παρακάτω αυστηρό τρόπο στην γραμμή εντολής (`tty`):
./stucs -l operationsfile -b hashentries -c config-file
όπου

- stucs είναι το εκτελέσιμο,
- operationsfile είναι το αρχείο που έχει τις εγγραφές που θα πρέπει να εισαχθούν, προσπελαστούν, η και να αλλαχτούν στις δομές της εφαρμογής.

  Τέτοιες λειτουργίες μπορούν να εισαχθούν και από το prompt της εφαρμογής.
- hashentries είναι ο μέγιστος αριθμός θέσεων για το πίνακα κατακερματισμού που θα δημιουργήσετε.
- config-file είναι ένα προαιρετικό configuration αρχείο που μπορείτε να το χρησιμοποιήσετε ώστε να παραμετροποιήσετε εσείς όπως επιθυμείτε την εφαρμογή σας.

Οι σημαίες -l/-b/-c μπορούν να χρησιμοποιηθούν με οποιαδήποτε σειρά στην γραμμή εκτέλεσης του προγράμματος και δεν μπορείτε να κάνετε αλλαγές στη ονομασία τους. Από τις παραπάνω in-line παραμέτρους μόνο -b είναι υποχρεωτική για την επιτυχή εκτέλεση του προγράμματος.

Περιγραφή της Διεπαφής του Προγράμματος:

Η εφαρμογή μέσω ενός prompt επιτρέπει στον χρήστη να αλληλεπιδρά με την δομή και να ανασύρει, αποθηκεύει, ή υπολογίζει διάφορες πληροφορίες με τις παρακάτω εντολές (η μορφή των εντολών είναι αυστηρή και θα πρέπει να χρησιμοποιηθούν όπως ακριβώς ορίζονται παρακάτω):

1. `i(nsert) studid lastname firstname gpa numcourses deprt postcode`: εισήγαγε στην δομή ένα φοιτητή με τα προφανή παραπάνω στοιχεία που ζει ταχυδρομικό κώδικα `postcode`. Το `studid` λειτουργεί σαν κλειδί για την εισαγωγή της εγγραφής στη δομή.

2. `q(uery) studid`: ανέσυρε και τύπωσε όλη την εγγραφή της φοιτήτριας με κλειδί `studid`.
3. `m(odify) studid gpa numcourses`: άλλαξε τον μέσο όρο και τον αριθμό των μαθημάτων που πρέπει να πάρει η φοιτήτρια `studid`.
4. `d(elete) studid`: διέγραψε από την δομή την εγγραφή που έχει σαν κλειδί το `studid`.
5. `ra(verage) studida studidb`: υπολογίστε το μέσο όρο των φοιτητών που έχουν αριθμό φοιτητικού μητρώου που βρίσκεται ανάμεσα στις τιμές `studida` και `studidb` (και συμπεριλαμβάνει τις τιμές αυτές).
6. `a(verage) postcode`: παρουσίασε μέσο όρο των φοιτητών που ζουν στον `postcode`.
7. `ta(verage) k postcode`: βρες και παρουσίασε τις `k` κορυφαίες φοιτήτριες που ζουν στον `postcode`.
8. `b(ottom) k`: παρουσίασε τις εγγραφές για τους `k` φοιτητές που έχουν τους μικρότερους μέσους όρους.
9. `ct(courses-to-take) postcode deprt`: παρουσίασε όλους του φοιτητές που ζουν στον `postcode` και σπουδάζουν στο τμήμα `deprt` καθώς και τον συνολικό αριθμό των μαθημάτων που οφείλουν.
10. `f(ind) gpa`: παρουσίασε όλους του σπουδαστές που οφείλουν το μέγιστό αριθμό από μαθήματα και ο μέσος όρος τους είναι μεγαλύτερος από `gpa`.
11. `p(ercentile) postcode`: υπολογίστε το ποσοστό των φοιτητών που ζουν στο κωδικό `postcode`.
12. `pe(rcentiles)`: παρουσιάστε όλους του ταχ. κωδικούς και για κάθε έναν το ποσοστό των φοιτητών από το σύνολικό αριθμό φοιτητών του ΕΚΠΑ που ζούν στον αντίστοιχο ταχ. κώδικα.
13. `e(xit)`: το πρόγραμμα απλά τερματίζει αφού ελευθερώσει πρώτα όλο το χώρο που έχει καταλάβει στην μνήμη.

Κάθε εγγραφή που αποθηκεύεται στην δομή(-ες) αποτελείται από:

- Αριθμός φοιτητή (μοναδικός ακέραιος αριθμός ή μοναδική σειρά χαρακτήρων)
- Επίθετο φοιτητή (σειρά χαρακτήρων)
- Όνομα φοιτητή (σειρά χαρακτήρων)
- Μέσος όρος (float με 2 ψηφία ακρίβειας).
- Αριθμός μαθημάτων (ακέραιος μεταξύ 0 και 52).
- Ταχυδρομικός κωδικός (σειρά 5 χαρακτήρων η ένας 5-ψήφιος ακέραιος)

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. Εκτός τις βασικές δομές μπορείτε να χρησιμοποιήσετε οποιαδήποτε άλλη δομή σας δίνει την δυνατότητα να απαντήσετε όλα τα ερωτήματα με πολυπλοκότητα *O(1)* και όσον αφορά τα ερωτήματα εύρους *O(κ)*. Θα πρέπει να περιγράψετε στην υποβληθείσα γραπτή εξήγηση πώς πετυχαίνετε κάτι τέτοιο.
3. Πριν να τερματίσει η εκτέλεση της εφαρμογής, το περιεχόμενο των δομών απελευθερώνεται συνολικά.
4. Αν πιθανώς κάποια κομμάτια του κωδικά σας προέλθουν από κάποια δημόσια πηγή, θα πρέπει *<u>να δώσετε αναφορά</u> 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 (1-2 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 `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. Προγράμματα που **δεν χρησιμοποιούν** separate compilation χάνουν αυτόματα 5% του βαθμού.
5. Σε καμιά περίπτωση τα Windows *δεν είναι επιλέξιμη* πλατφόρμα για την παρουσίαση αυτής της άσκησης.

Αναφορές

[1] William Pugh, Skip Lists: A Probabilistic Alternative to Balanced Trees *Communications of the ACM*, vol. 33, no. 6, pp. 668–676, June 1990. *Available at*: `ftp://ftp.cs.umd.edu/pub/skipLists/skiplists.pdf`

[2] Alex Delis, A Brief on Skip-Lists, *Technical Note*, Univ. of Athens, Athens, Greece, September 2016, *Available at*: `http://cgi.di.uoa.gr/~ad/k22/skiplist-note.pdf`.