Γενική περιγραφή

Θα υλοποιήσετε μια εφαρμογή αξιολόγησης κριτικών για ταινίες. Η εφαρμογή σας θα μελετήσει μια συλλογή κριτικών οι οποίες έχουν ήδη αξιολογηθεί και με βάση τα στοιχεία που θα συλλέξει θα μπορεί να αξιολογήσει νέες κριτικές που εισάγονται από το πληκτρολόγιο.

Τα δεδομένα που θα χρησιμοποιήσει η εφαρμογή σας είναι μια συλλογή από μερικές χιλιάδες κριτικές ταινιών κάθε μία εκ των οποίων έχει ήδη αξιολογηθεί με έναν ακέραιο 0-4 με σημασία:

0 - αρνητική
1 - κάπως αρνητική
2 - ουδέτερη
3 - κάπως θετική
4 - θετική

Τα δεδομένα παρέχονται σε αρχείο κειμένου. Κάθε γραμμή του αρχείου περιέχει το βαθμό αξιολόγησης μιας κριτικής και μετά το κείμενο αυτής της κριτικής. Θα σας δώσουμε μια έτοιμη συνάρτηση η οποία σε κάθε κλήση της επιστρέφει την επόμενη γραμμή του αρχείου.

Το πρόγραμμά σας θα πρέπει να αποσπά τις λέξεις κάθε κριτικής και να τις αποθηκεύει σε μια δομή δεδομένων που θα σας περιγράψουμε μαζί με ένα σκορ που προκύπτει από το βαθμό αξιολόγησης της κριτικής.

Αφού συλλεχθούν όλα τα δεδομένα, το πρόγραμμα ζητά από το χρήστη να εισάγει μια νέα, μη-αξιολογημένη κριτική και με βάση τα σκορ που έχει ήδη υπολογίσει να την αξιολογεί με βάση την παραπάνω κλίμακα.

Επιπλέον, το πρόγραμμα πρέπει να υπολογίζει αναδρομικά τη λέξη που έχει το καλύτερο σκορ από όλες όσες έχουν ήδη εισαχθεί στη δομή δεδομένων.

Δομές δεδομένων

Η βασική δομή δεδομένων είναι ένας πίνακας κατακερματισμού (hash table) υλοποιημένος ως πίνακας λιστών. Το βασικό χαρακτηριστικό ενός hash table είναι ότι χρησιμοποιούμε μια ειδική συνάρτηση (hash function) για να βρούμε σε ποια θέση του πίνακα θα αποθηκευτεί κάθε στοιχείο. Στοιχεία που αντιστοιχίζονται στην ίδια θέση εισάγονται στη λίστα που βρίσκεται στη θέση αυτή. Αν η hash function είναι "καλή", τότε μοιράζει ομοιόμορφα τα στοιχεία στον πίνακα, με αποτέλεσμα οι λίστες να μην είναι ιδιαίτερα μεγάλες και η αναζήτηση σε αυτές να είναι πολύ γρήγορη.

Ορίζουμε ως load factor την ποσότητα (πλήθος στοιχείων στο hash table) / (μέγεθος πίνακα). Οσο πιο μεγάλο είναι το load factor, τόσο πιο μεγάλες είναι δυνητικά οι λίστες. Συνήθως, όταν το load factor υπερβαίνει κάποιο όριο επιλέγουμε να μεγαλώσουμε τον πίνακα και να ξανατοποθετήσουμε τα ήδη υπάρχοντα στοιχεία στο νέο πίνακα. Η διαδικασία αυτή λέγεται rehashing. Σημειώστε πως τα στοιχεία θα μπουν σε διαφορετικές θέσεις γιατί η θέση ενός στοιχείου εξαρτάται και από το μέγεθος του πίνακα.

Στα πλαίσια της εργασίας θα πρέπει να υλοποιήσετε:

Δομή 1: Διασυνδεδεμένη λίστα

Ενας κόμβος της λίστας θα αντιστοιχεί σε ένα στοιχείο που αποθηκεύεται στον hash table και, πέρα από δείκτες προς άλλους κόμβους, θα πρέπει να περιέχει:

- τη λέξη, ως char *. Εννοείται δυναμικά δεσμευμένη μνήμη για τη λέξη.
- πλήθος εμφανίσεων της λέξης σε κριτικές.
- αθροιστικό σκορ της λέξης.

Είστε ελεύθεροι να υλοποιήσετε τη λίστα όπως πιστεύετε είναι πιο βολικό, δεδομένου ότι θα χρησιμοποιηθεί στο hashtable. Διαβάστε προσεκτικά την εκφώνηση και κυρίως τις λειτουργίες που έχουν σχέση με τα στοιχεία που αποθηκεύονται στον πίνακα κι αποφασίστε αν σας διευκολύνει η λίστα να έχει τερματικό ή όχι, να είναι διπλά διασυνδεδεμένη ή όχι, να είναι κυκλική ή όχι. Θα χρειαστείτε τουλάχιστον τις παρακάτω συναρτήσεις:

- συνάρτηση η οποία, δεδομένης μιας λέξης και σκορ, είτε εισάγει τη λέξη στη λίστα αν αυτή δεν υπάρχει ήδη ή ανανεώνει τις εμφανίσεις και το σκορ της, αν ήδη υπάρχει.
- συνάρτηση που εκτυπώνει τα στοιχεία της λίστας (με τρόπο που θα περιγράψουμε σε άλλη ενότητα)
- συνάρτηση που καταστρέφει τη λίστα
- συνάρτηση η οποία, δεδομένης μιας λέξης, βρίσκει, αφαιρεί, κι επιστρέφει (χωρίς να καταστρέψει) τον κόμβο της λίστας που αντιστοιχεί σε αυτή τη λέξη, εφόσον υπάρχει.

Καλό είναι να έχετε κι άλλες συναρτήσεις πέρα από τις προτεινόμενες.

Δομή 2: Hash table

Κατασκευάστε ένα struct της μορφής:

```
typedef struct {
    int size;
    int num_entries;
    entryT *table;
} hashT;
```

όπου:

table είναι ο hash table: ένας δυναμικά δεσμευμένος πίνακας από κεφαλές λιστών.

size είναι το μέγεθος του hash table

num_entries είναι το πλήθος στοιχείων που έχουν αποθηκευτεί στον πίνακα.

Ουσιαστικά το struct ομαδοποιεί τα στοιχεία που συγκεντρωτικά αποτελούν μια δομή hash table.

Ορίστε μια καθολική μεταβλητή τύπου hashT την οποία θα μπορείτε να χρησιμοποιήσετε στο πρόγραμμά σας για να προσπελάσετε τον hash table. Αυτή είναι η μοναδική καθολική μεταβλητή που επιτρέπεται να έχετε στο πρόγραμμα.

Θα χρειαστείτε τουλάχιστον τις παρακάτω συναρτήσεις:

- συνάρτηση η οποία δεδομένης μιας λέξης και του σκορ της, βρίσκει σε ποια θέση (με άλλα λόγια σε ποια λίστα) πρέπει να εισαχθεί η λέξη και χρησιμοποιεί τη συνάρτηση εισαγωγής σε λίστα ώστε να εισάγει ή να ανανεώσει το αντίστοιχο στοιχείο.
- συνάρτηση εκτύπωσης των περιεχομένων του hash table (με τρόπο που θα περιγράψουμε σε άλλη ενότητα)
- συνάρτηση που υλοποιεί τη λειτουργία rehash
- συνάρτηση που καταστρέφει το hash table
- τη hash function η οποία παίρνει μια λέξη και μας επιστρέφει έναν ακέραιο. Το υπόλοιπο της διαίρεσης αυτού του ακεραίου δια του μεγέθους του πίνακα μας δίνει τη θέση στον πίνακα όπου πρέπει να αποθηκευτεί η λέξη. Χρησιμοποιήστε την παρακάτω hash function:

```
unsigned long hash(char *str) {

    unsigned long hash = 5381;
    int c;

    while ((c = *str++))
        hash = ((hash << 5) + hash) + c;

    return hash;
}
```

Λειτουργία προγράμματος

Ακολουθούν λεπτομέρειες για το πώς ακριβώς πρέπει να λειτουργεί το πρόγραμμά σας. Εχουμε χωρίσει τη διαδικασία σε βήματα για τη διευκόλυνσή σας.

Βήμα 1: Επεξεργασία γραμμής εντολής

Ο χρήστης δίνει τις παρακάτω πληροφορίες στη γραμμή εντολής:

- το όνομα του αρχείου δεδομένων
- (προαιρετικά) την επιλογή `-p`

Αν δεν έχει δοθεί σωστό πλήθος παραμέτρων στη γραμμή εντολής, το πρόγραμμα τερματίζει αφού εκτυπώσει το μήνυμα Incorrect number of parameters. ακολουθούμενο από χαρακτήρα αλλαγής γραμμής.

Αν έχει δοθεί η επιλογή -p τότε σε επόμενο στάδιο θα εκτυπωθούν τα περιεχόμενα του hash table, διαφορετικά δεν εκτυπώνονται.

Βήμα 2: Εισαγωγή δεδομένων

Κατασκευάστε κι αρχικοποιήστε τον hash table. Δώστε του αρχικό μέγεθος 100.

Θα σας δώσουμε τη συνάρτηση char* read_next_line (char *filename); η οποία παίρνει ως παράμετρο το όνομα του αρχείου δεδομένων και επιστρέφει δυναμικά δεσμευμένη συμβολοσειρά η οποία περιέχει μια γραμμή του αρχείου. Κάθε κλήση στη συνάρτηση επιστρέφει την επόμενη γραμμή. Οταν δεν υπάρχουν άλλες γραμμές, επιστρέφει NULL.

Για κάθε γραμμή που διαβάζετε:

- αποσπάτε την πρώτη "λέξη" η οποία είναι πάντα μια ακέραια τιμή που εκφράζει το σκορ αξιολόγησης της κριτικής.
- αποσπάτε μία-μία τις επόμενες λέξεις. Για κάθε λέξη,
  - μετατρέψτε τη σε μικρά γράμματα
  - βρείτε τη θέση στο hash table στην οποία αντιστοιχεί αυτή η λέξη και ελέγξτε αν υπάρχει ήδη στην αντίστοιχη λίστα.
    - Αν δεν υπάρχει, τότε την προσθέτετε στη λίστα, με σκορ το σκορ αξιολόγησης της κριτικής
      - Μετά την εισαγωγή νέας λέξης, ελέγξτε αν το load factor του πίνακα έχει υπερβεί την τιμή 3. Αν ναι, τότε υλοποιήστε τη λειτουργία rehash: κατασκευάστε ένα νέο hash table διπλάσιου μεγέθους από τον προηγούμενο. Διατρέξτε τον αρχικό πίνακα και για κάθε κόμβο λίστας που βρίσκετε, αφαιρέστε τον κόμβο (χωρίς να τον καταστρέψετε) και εισάγετέ τον στη θέση του νέου πίνακα την οποία σας δίνει το hash function. Η εισαγωγή πρέπει να γίνει προσαρμόζοντας αναλόγως τους δείκτες προς τον επόμενο κι (αν η λίστα είναι διπλή) προηγούμενο κόμβο. Μην ξανακάνετε malloc! Σημειώστε πως εφόσον έχει αλλάξει το μέγεθος του πίνακα, η hash function θα δίνει διαφορετικές τιμές από ότι πριν.
    - Αν υπάρχει ήδη, τότε αυξήστε το πλήθος εμφανίσεων της λέξης και προσθέστε το σκορ αξιολόγησης της κριτικής στο σκορ της λέξης.

Κάποιες "λέξεις" θα είναι σημεία στίξης. Δε χρειάζεται να κάνετε έλεγχο γι αυτά. Απλά εισάγετέ τα κανονικά στο hash table.

Βήμα 3: Αναδρομική εύρεση "καλύτερης" λέξης.

Για κάθε λέξη έχουμε καταγράψει το αθροιστικό σκορ αξιολόγησής της και το πλήθος εμφανίσεών της. Το πηλίκο αυτών των τιμών μας δίνει το μέσο σκορ της λέξης και είναι μια ένδειξη του πόσο "θετική" είναι. Σε αυτό το βήμα θα βρείτε με αναδρομικό τρόπο τη λέξη που έχει το μεγαλύτερο μέσο σκορ.

Βήμα 3α. Κατασκευάστε ένα πίνακα μεγέθους όσο ο hash table. Σε κάθε θέση του πίνακα θέλουμε να αποθηκευτεί μια λέξη και το μέσο σκορ της (ή εναλλακτικά το αθροιστικό σκορ και το πλήθος εμφανίσεων). Μπορείτε να χρησιμοποιήσετε το ίδιο struct που φτιάξατε για τους κόμβους της λίστας. Ας υποθέσουμε ότι έχετε ονομάσει αυτό το struct entryT.

Γράψτε μια συνάρτηση η οποία παίρνει ως παράμετρο την κεφαλή μιας λίστας, αναζητά σε αυτή ***αναδρομικά*** τη λέξη με το μεγαλύτερο μέσο σκορ και επιστρέφει είτε NULL (αν η λίστα είναι άδεια) είτε δείκτη στο entryT που περιέχει τη λέξη που βρέθηκε.

Τρέξτε αυτή τη συνάρτησή σας για κάθε θέση του hash table και αποθηκεύστε τα αποτελέσματα στον πίνακα που φτιάξατε στην αρχή αυτού του βήματος.

Βήμα 3β. Γράψτε μια συνάρτηση η οποία παίρνει ως παραμέτρους τον πίνακα του βήματος 3α και έναν ακέραιο που αναπαριστά θέση του πίνακα, αναζητά σε αυτόν ***αναδρομικά*** τη λέξη με το μεγαλύτερο μέσο σκορ και επιστρέφει το αντίστοιχο entryT.

Βήμα 3γ. Το πρόγραμμά σας εκτυπώνει το μήνυμα The most positive word is "W" with a score of S.SSS ακολουθούμενο από χαρακτήρα αλλαγής γραμμής. W είναι η λέξη και S.SSS είναι το μέσο σκορ της με τρία δεκαδικά ψηφία.

Βήμα 4: Εκτύπωση hash table

Αν είχε δοθεί η παράμετρος προγράμματος -p, τότε σε αυτό το σημείο εκτυπώνετε τα περιεχόμενα του hash table με χαρακτήρα αλλαγής γραμμής πριν και μετά. Για την ακρίβεια,

για κάθε θέση του πίνακα που περιέχει μη-άδεια λίστα, εκτυπώνετε:

- τη θέση του πίνακα με πλάτος 4
- άνω-κάτω τελεία
- ένα κενό
- τα περιεχόμενα της λίστας

για κάθε λίστα εκτυπώνετε:

- τα περιεχόμενα κάθε κόμβου (εκτός πιθανού τερματικού κόμβου)
- κόμμα ακολουθούμενο από ένα κενό, αν ο κόμβος δεν είναι ο τελευταίος
- χαρακτήρα αλλαγής γραμμής, αν ο κόμβος είναι ο τελευταίος.

για κάθε κόμβο εκτυπώνετε:

- [
- κενό
- "W" όπου W η λέξη
- κενό
- πλήθος εμφανίσεων της λέξης
- κενό
- αθροιστικό σκορ της λέξης με δύο δεκαδικά ψηφία.
-]

Βήμα 5

Σε επανάληψη μέχρις ότου ο χρήστης εισάγει τη λέξη DONE:

- Εκτυπώστε χαρακτήρα αλλαγής γραμμής, το μήνυμα Enter review or DONE to finish: και χαρακτήρα αλλαγής γραμμής.
- Διαβάστε μια γραμμή από το πληκτρολόγιο. Υποθέστε ότι η γραμμή θα είναι μια πλήρης κριτική ταινίας κι ότι πιθανά σημεία στίξης θα έχουν κενά γύρω τους ώστε να μην είναι "κολλημένα" σε λέξεις.
- Για κάθε μία λέξη βρείτε από το hash table το μέσο σκορ της. Το σκορ αξιολόγησης της κριτικής είναι το άθροισμα των μέσων σκορ των λέξεών της που εμφανίζονται στον hash table δια του πλήθους αυτών.
- Αν το πλήθος λέξεων είναι 0, εκτυπώστε το μήνυμα Sorry, there is no score for this review! ακολουθούμενο από χαρακτήρα αλλαγής γραμμής. Αυτό θα έχει συμβεί αν ο χρήστης απλά πάτησε enter όταν ζητήθηκε να εισάγει την κριτική του.
- Διαφορετικά,
    - εκτυπώστε το μήνυμα Review score: S.SSSS ακολουθούμενο από χαρακτήρα αλλαγής γραμμής, όπου S.SSSS το σκορ αξιολόγησης της κριτικής, με 4 δεκαδικά ψηφία.
    - εκτυπώστε το μήνυμα This review is (υπάρχει ένα κενό μετά το is) κι ανάλογα με το αν το σκορ αξιολόγησης είναι μικρότερο, ίσο ή μεγαλύτερο του 2 τη λέξη negative, neutral ή positive αντίστοιχα. Ακολουθεί τελεία και χαρακτήρας αλλαγής γραμμής.

Βήμα 6

Αποδεσμεύστε όλη τη δυναμικά δεσμευμένη μνήμη του προγράμματος.

Έλεγχος ορθότητας

Χρησιμοποιήστε τα ενδεικτικά αρχεία εισόδου/εξόδου που θα σας δώσουμε για να ελέγξετε την ορθότητα της εφαρμογής σας. Η έξοδός σας πρέπει να ταιριάζει ακριβώς με τη δική μας.