HY240: Δομές Δεδομένων
Εαρινό Εξάμηνο – Ακαδημαϊκό Έτος 2016
Διδάσκουσα: Παναγιώτα Φατούρου

Προγραμματιστική Εργασία
2ο Μέρος

Ημερομηνία Παράδοσης: Κυριακή, 15 Μαίου 2016, ώρα 23:59.

Τρόπος Παράδοσης: Χρησιμοποιώντας το πρόγραμμα turnin. Πληροφορίες για το πώς λειτουργεί το turnin παρέχονται στην ιστοσελίδα του μαθήματος.

Γενική Περιγραφή

Στην εργασία αυτή καλείστε να υλοποιήσετε ένα πρόγραμμα που προσομοιώνει μια υπηρεσία πληροφοριών σχετικά με ταινίες. Η υπηρεσία διαθέτει ένα σύνολο ταινιών ταξινομημένες σε θεματικές κατηγορίες. Οι χρήστες μπορούν να εγγράφονται στην υπηρεσία ώστε να βαθμολογούν ταινίες που έχουν ήδη δεί. Η υπηρεσία μπορεί επιπρόσθετα να προτείνει ταινίες στους χρήστες σύμφωνα με τις προτιμήσεις τους βάσει των ταινιών που έχουν ήδη δει.

Αναλυτική Περιγραφή Ζητούμενης Υλοποίησης

Δομές δεδομένων που αφορούν τις ταινίες

Η υπηρεσία διαθέτει μια συλλογή ταινιών ταξινομημένες σε κατηγορίες. Πιο συγκεκριμένα, οι διαθέσιμες κατηγορίες ταινιών είναι οι παρακάτω: Δραματικές, Oscar, Cinephile, Ντοκιμαντέρ, Κινούμενα σχέδια.

Για τη διαχείριση αυτών των κατηγοριών, θα δημιουργήσετε έναν **πίνακα σταθερού μεγέθους 5 θέσεων**, μια για κάθε κατηγορία, ο οποίος θα ονομάζεται *πίνακας κατηγοριών*.

Κάθε θέση του πίνακα θα περιέχει δύο δείκτες και έναν ακέραιο. Ο πρώτος δείκτης θα δείχνει σε **ένα απλά συνδεδεμένο δένδρο δυαδικής αναζήτησης (binary search tree), το οποίο είναι ταξινομημένο ως προς το πεδίο movieID των κόμβων του και έχει κόμβο φρουρό.** Το δένδρο αυτό ονομάζεται *δένδρο ταινιών της κατηγορίας*. Ο δεύτερος δείκτης θα δείχνει στον κόμβο φρουρό του δένδρου. Ο ακέραιος **(movieCounter)** αποθηκεύει το συνολικό αριθμό ταινιών που υπάρχουν στην κατηγορία (δηλαδή το πλήθος των κόμβων του δένδρου ταινιών της κατηγορίας αυτής).

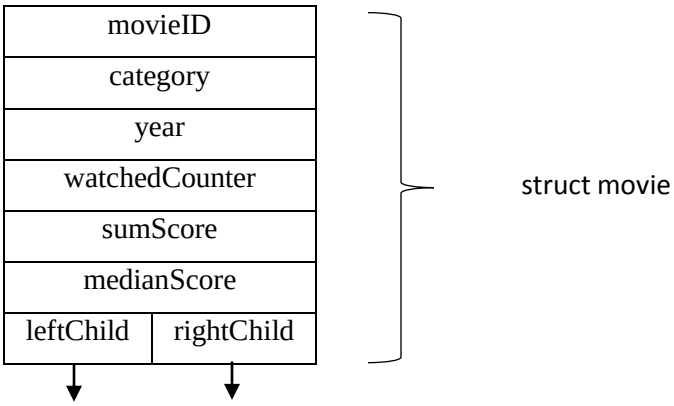
Κάθε στοιχείο του δένδρου ταινιών μιας συγκεκριμένης κατηγορίας είναι ένα struct τύπου movie με τα εξής πεδία:

- Έναν ακέραιο **movieID** που αποτελεί το μοναδικό αναγνωριστικό της ταινίας.
- Έναν ακέραιο **category** που αντιστοιχεί στη θεματική κατηγορία της ταινίας. Η μεταβλητή αυτή λαμβάνει τιμές από 0 έως 4, όπου η αντιστοίχιση γίνεται ως ακολούθως: 0: Δραματική, 1: Oscar, 2: Cinephile, 3: Ντοκιμαντέρ, 4: Κινούμενα Σχέδια
- Έναν ακέραιο **year** που αντιστοιχεί στο έτος κυκλοφορίας της ταινίας.
- Έναν ακέραιο **watchedCounter** που αντιστοιχεί στον αριθμό των χρηστών που έχουν δει την ταινία.
- Έναν ακέραιο **sumScore** που αντιστοιχεί στο άθροισμα της βαθμολογίας που έχουν δώσει οι χρήστες για την ταινία.
- Έναν αριθμό κινούμενης υποδιαστολής (float) **medianScore**, που αντιστοιχεί στη μέση βαθμολογία της ταινίας.
- Ένα δείκτη **leftChild** που δείχνει στον αριστερό θυγατρικό κόμβο του κόμβου που αντιστοιχεί στην ταινία.
- Ένα δείκτη **rightChild** που δείχνει στον δεξιό θυγατρικό κόμβο του κόμβου που αντιστοιχεί στην ταινία.

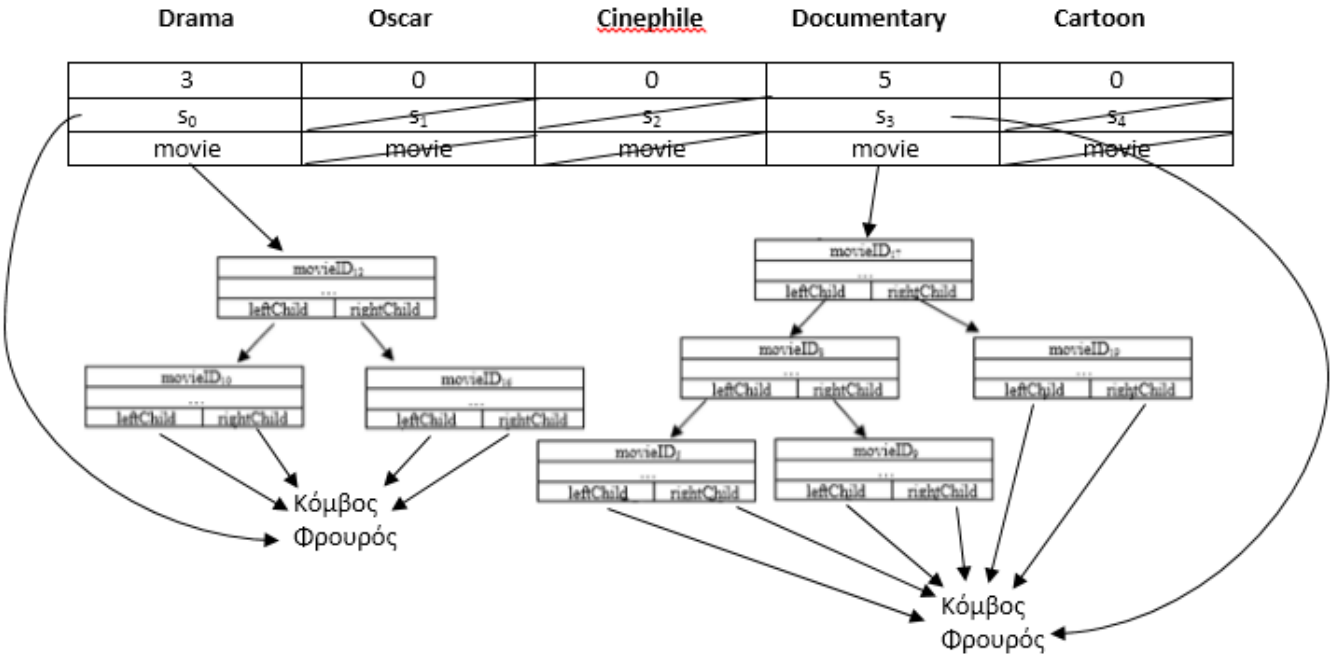
Μια εγγραφή του τύπου movie παρουσιάζεται στο Σχήμα 1.

Επιπρόσθετα, σε αυτή τη φάση της εργασίας, οι ταινίες που χαρακτηρίζονται ως *«νέες κυκλοφορίες»* θα διατηρούνται σε ένα επιπρόσθετο δένδρο (ανεξάρτητο από τα δένδρα που δεικτοδοτούνται από τα στοιχεία του πίνακα κατηγοριών), κάθε κόμβος του οποίου είναι επίσης ένα struct τύπου movie. Το δένδρο αυτό ονομάζεται δένδρο *νέων κυκλοφοριών* και είναι ένα **απλά συνδεδεμένο δένδρο, ταξινομημένο βάσει του πεδίου movieID των κόμβων του. Σε αντίθεση με το δένδρο ταινιών μιας κατηγορίας, το δένδρο νέων κυκλοφοριών δεν έχει κόμβο φρουρό.**

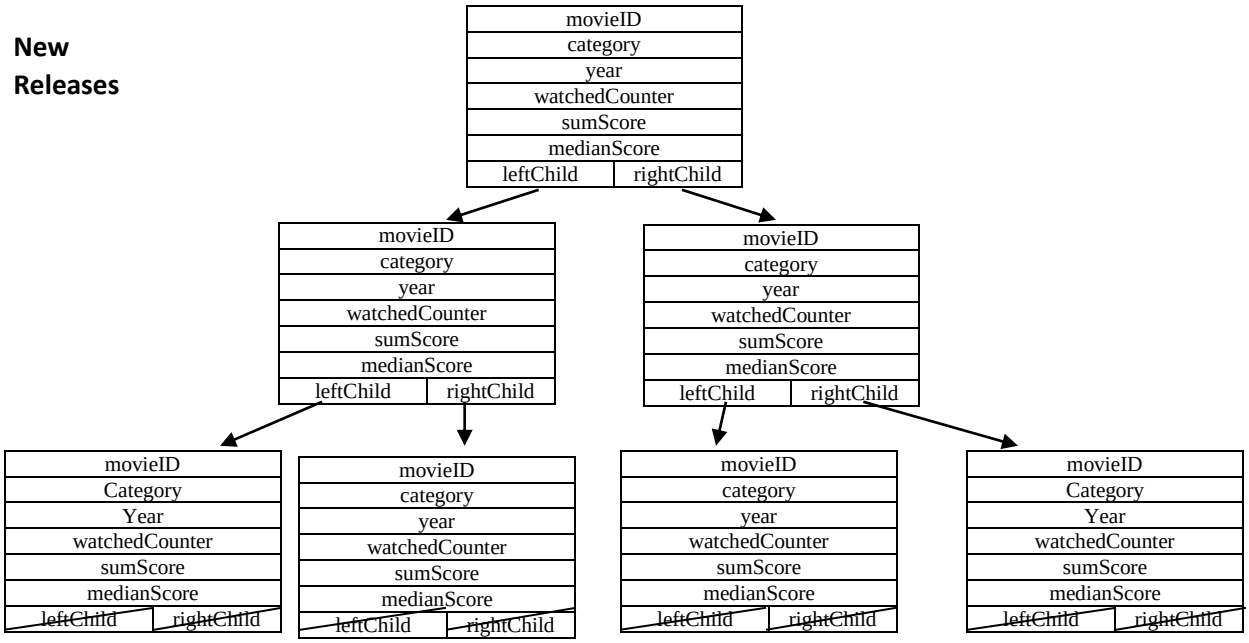
Στο Σχήμα 2 (2α και 2β) παρουσιάζεται ο πίνακας κατηγοριών και το δένδρο ταινιών που δεικτοδοτείται από κάθε στοιχείο του. Στο Σχήμα 3 παρουσιάζεται πιο αναλυτικά ένα δένδρο ταινιών μιας συγκεκριμένης κατηγορίας.



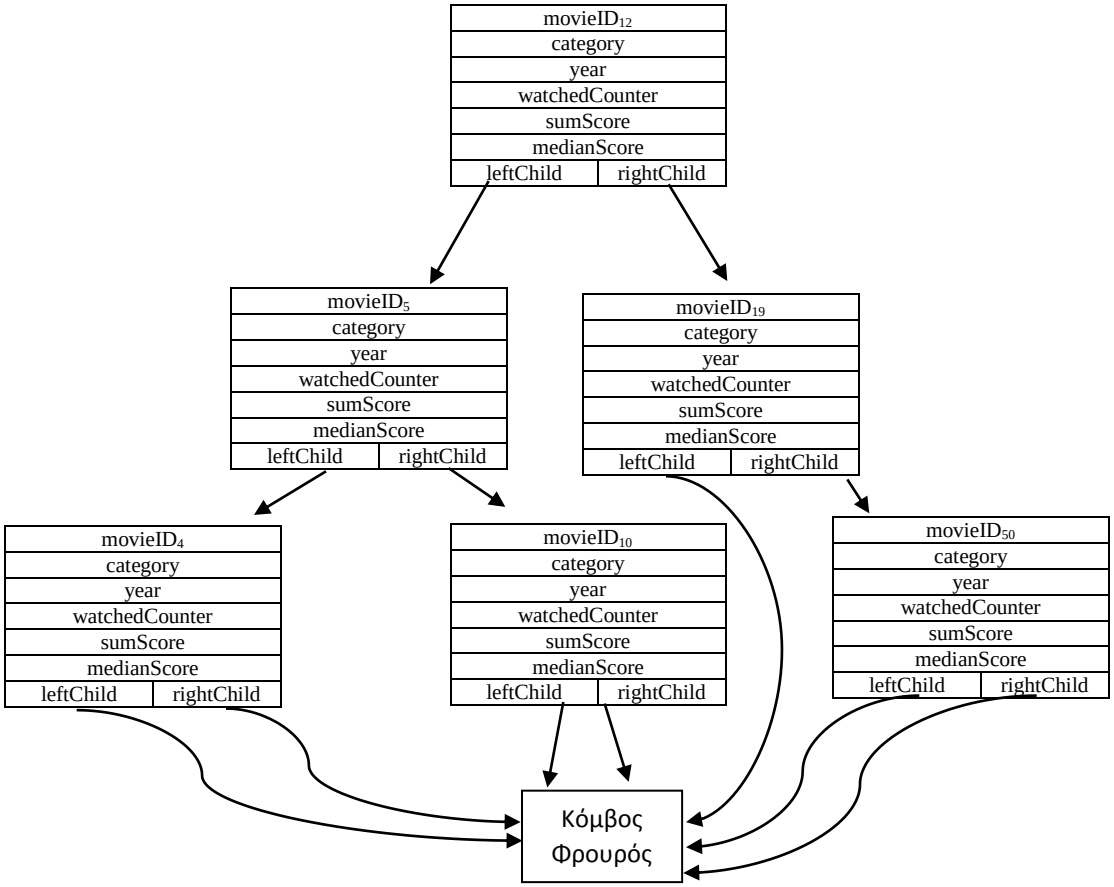
Σχήμα 1: Εγγραφή τύπου movie



Σχήμα 2α: Πίνακας Κατηγοριών και δένδρα ταινιών κατηγοριών



Σχήμα 2β: Δένδρο νέων κυκλοφοριών



Σχήμα 3: Απλά συνδεδεμένο δυαδικό δένδρο συγκεκριμένης κατηγορίας

Δομές δεδομένων που αφορούν τον χρήστη

Η υπηρεσία εξυπηρετεί ένα σύνολο εγγεγραμμένων χρηστών. Οι χρήστες σε αυτή τη φάση της εργασίας θα διατηρούνται σε έναν **πίνακα κατακερματισμού USER[hash_table_size],** ο οποίος περιέχει πληροφορίες για τους χρήστες.

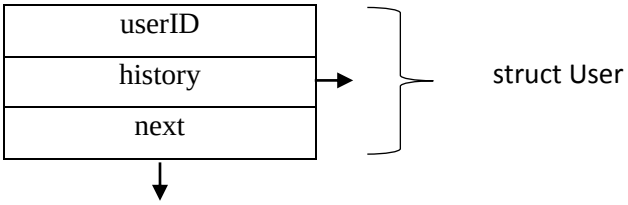
Το μέγεθος του πίνακα κατακερματισμού **hash_table_size** θα πρέπει να επιλέγεται από εσάς προσεκτικά και θα πρέπει να είστε σε θέση να δικαιολογήσετε την επιλογή σας.

Κάθε στοιχείο του USER[i] του πίνακα αυτού περιέχει έναν δείκτη στο πρώτο στοιχείο **μιας απλά συνδεδεμένης αλυσίδας. Κάθε στοιχείο μιας αλυσίδας είναι μια εγγραφή (struct), τύπου user, με τα εξής πεδία:**

- Έναν ακέραιο, **userID,** που χαρακτηρίζει μοναδικά το χρήστη.
- Ένα δείκτη, **history**, σε ένα struct τύπου userMovie (βλέπε παρακάτω), που δείχνει σε ένα **διπλά-συνδεδεμένο φυλλο-προσανατολισμένο δένδρο δυαδικής αναζήτησης,** το οποίο ονομάζεται *δένδρο ιστορικού ταινιών του χρήστη***. Το δένδρο αυτό είναι ταξινομημένο** ως προς το πεδίο movieID και περιέχει ταινίες που έχει ήδη παρακολουθήσει και βαθμολογήσει ο χρήστης.
- Ένα δείκτη, **next,** που δείχνει στο επόμενο στοιχείο της λίστας χρηστών.

Η κάθε αλυσίδα του πίνακα κατακερματισμού είναι **ταξινομημένη κατ' αύξουσα διάταξη βάσει του πεδίου userID** των κόμβων της. Προσέξτε ότι το userID κάθε χρήστη της αλυσίδας που δεικτοδοτείται από τη θέση i του πίνακα κατακερματισμού, έχει τιμή κατακερματισμού i.

Η εγγραφή τύπου User παρουσιάζεται στο Σχήμα 4.



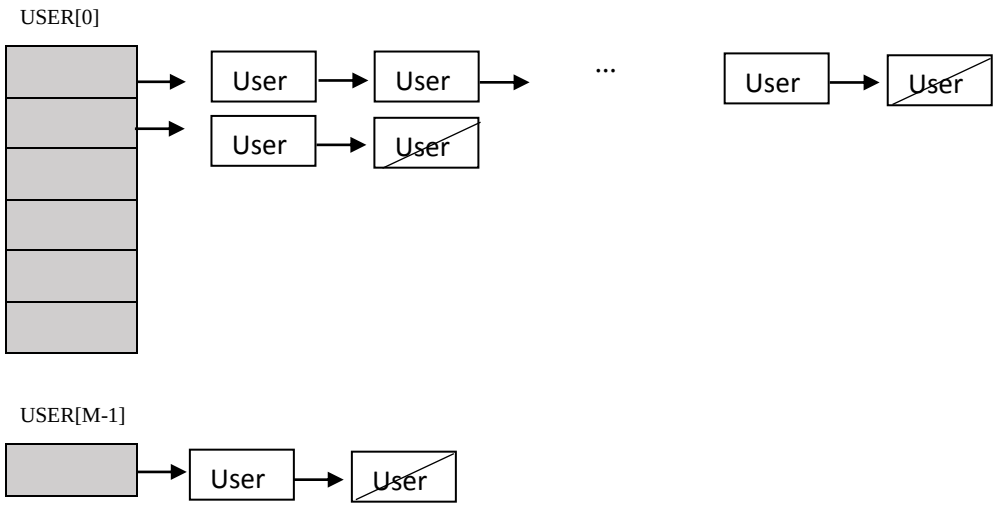
Σχήμα 4: Εγγραφή τύπου user

Για την υλοποίηση της συνάρτησης κατακερματισμού θα πρέπει να βασιστείτε στην τεχνική του καθολικού κατακερματισμού. Για την υλοποίηση του καθολικού κατακερματισμού θα δίνονται τα εξής:

1) Ένας πίνακας primes[], ο οποίος περιέχει πρώτους αριθμούς σε αύξουσα σειρά.
2) Το μέγιστο πλήθος χρηστών, μέσω της μεταβλητής max_users
3) Το μέγιστο αναγνωριστικό χρήστη userID, μέσω της μεταβλητής max_id

Αυτές οι μεταβλητές είναι global, έχουν δηλωθεί στο αρχείο Movie.h **και θα αρχικοποιούνται στη main βάσει τιμών που αναγράφονται στις πρώτες γραμμές κάθε test_file.**

Για την επίλυση των συγκρούσεων θα ακολουθήσετε την μέθοδο των **ταξινομημένων αλυσίδων.** Ο πίνακας κατακερματισμού χρηστών παρουσιάζεται στο Σχήμα 5.

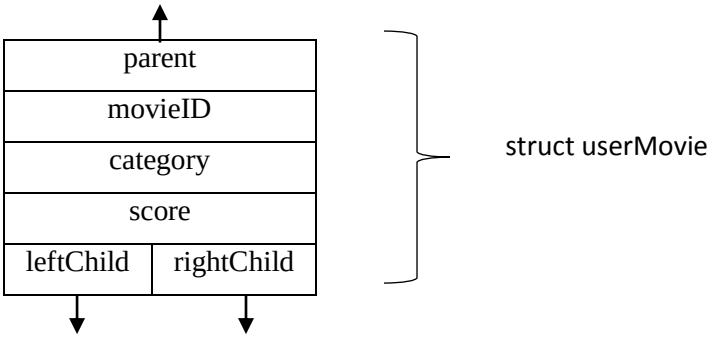


Σχήμα 5: Πίνακας κατακερματισμού χρηστών χρησιμοποιώντας την μέθοδο ταξινομημένων αλυσίδων

Ο κάθε κόμβος του δένδρου ιστορικού ταινιών του κάθε χρήστη αντιστοιχεί σε μια εγγραφή τύπου userMovie. Το struct userMovie περιέχει:

- Έναν ακέραιο **movieID** που χαρακτηρίζει μοναδικά την ταινία
- Έναν ακέραιο **category** που αντιστοιχεί στη θεματική κατηγορία της ταινίας. Η μεταβλητή αυτή λαμβάνει τιμές από 0 εώς 4, όπου 0: drama, 1: Oscar, 2: cinephile, 3: documentary, 4: cartoon.
- Έναν ακέραιο **score** που αντιπροσωπεύει την βαθμολογία που έδωσε ο χρήστης στη ταινία. Η μεταβλητή αυτή παίρνει τιμές στο διάστημα 1 εως 10, με 1 να είναι η χαμηλότερη βαθμολογία για μια ταινία και 10 η μεγαλύτερη.
- Έναν δείκτη **parent** που δείχνει στον πατέρα του κομβού
- Έναν δείκτη **leftChild** που δείχνει στον αριστερό θυγατρικό κόμβο.
- Έναν δείκτη **rightChild** που δείχνει στον δεξιό θυγατρικό κόμβο.

Η εγγραφή τύπου userMovie παρουσιάζεται στο Σχήμα 6.

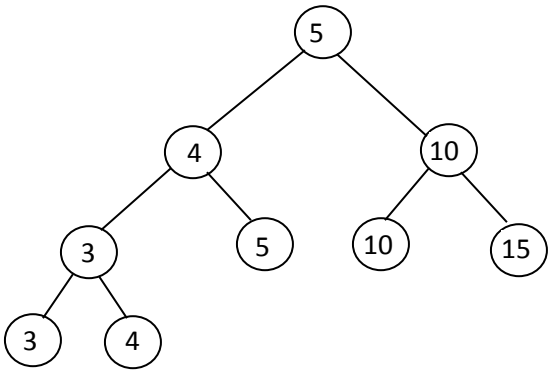


Σχήμα 6: Εγγραφή τύπου userMovie

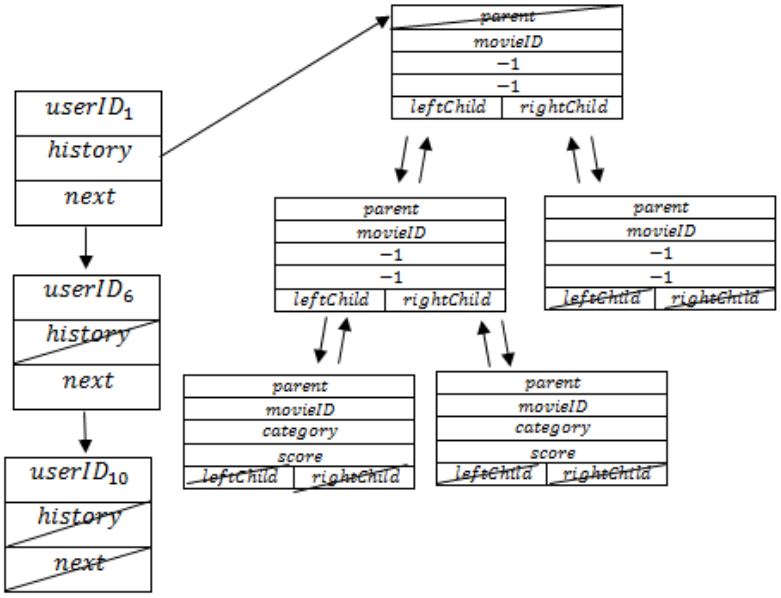
Το *δένδρο ιστορικού* είναι **διπλά-συνδεδεμένο φυλλο-προσανατολισμένο δένδρο δυαδικής αναζήτησης**. Τα φυλλοπροσανατολισμένα δένδρα δυαδικής αναζήτησης (leaf-oriented binary search trees) αποτελούν μια εναλλακτική υλοποίηση του αφηρημένου τύπου δεδομένων του λεξικού. Ορίζονται ως εξής: (α) Όλα τα κλειδιά του λεξικού αποθηκεύονται στα φύλλα του δένδρου, από αριστερά προς τα δεξιά κατά μη φθίνουσα τιμή κλειδιού, και (β) οι εσωτερικοί κόμβοι αποθηκεύουν κλειδιά (που δεν αντιστοιχούν απαραίτητα σε κλειδιά του λεξικού), έτσι ώστε να ισχύει η κάτωθι αμετάβλητη συνθήκη σε κάθε κόμβο v:

Το κλειδί του αριστερού παιδιού του v είναι μικρότερο ή ίσο από αυτό του v,
ενώ το δεξιό παιδί του v διαθέτει κλειδί μεγαλύτερο από εκείνο του v

Παρατηρήστε ότι βάσει του ορισμού, οι εσωτερικοί κόμβοι έχουν και τους δύο δείκτες μη κενούς, ενώ και οι δύο δείκτες των φύλλων είναι κενοί. Ένα παράδειγμα φυλλο-προσανατολισμένου δένδρου παρουσιάζεται στο παρακάτω σχήμα. Οι λειτουργίες της εισαγωγής και της διαγραφής περιγράφηκαν στην 3η σειρά θεωρητικών ασκήσεων του μαθήματος.



Στο Σχήμα 7 παρουσιάζεται το δένδρο ιστορικού ενός χρήστη που δεικτοδοτείται από κάποιον από τους κόμβους μιας αλυσίδα του πίνακα κατακερματισμού.



Σχήμα 7: Παράδειγμα μιας αλυσίδας χρηστών και του διπλά συνδεδεμένου φυλλο-προσανατολισμένου δένδρου ιστορικού

Τρόπος Λειτουργίας Προγράμματος

Το πρόγραμμα που θα δημιουργηθεί θα πρέπει να εκτελείται καλώντας την ακόλουθη εντολή:

<executable> <input-file>

όπου <executable> είναι το όνομα του εκτελέσιμου αρχείου του προγράμματος (π.χ. a.out) και <input-file> είναι το όνομα ενός αρχείου εισόδου (π.χ. testfile) το οποίο περιέχει τα γεγονότα.

Τα γεγονότα εισόδου είναι τα εξής:

- **R <userID >**

  Γεγονός τύπου *register user* το οποίο σηματοδοτεί την εγγραφή ενός νέου χρήστη (user) με αναγνωριστικό <userID>. Το γεγονός αυτό προσθέτει το νέο χρήστη στον πίνακα κατακερματισμού χρηστών της υπηρεσίας. Το πεδίο history του χρήστη πρέπει να έχει την αρχική τιμή NULL.

  Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
R <userID>
Chain <j> of Users:
    <userID₁>
    <userID₂>
       ...
    <userIDₙ>
DONE
```

  όπου <j> είναι η τιμή κατακερματισμού του κλειδιού <userID>, n είναι ο αριθμός των χρηστών στην αλυσίδα που δεικτοδοτείται από τη θέση <j> του πίνακα Users και για κάθε $i \in \{1, \ldots, n\}$, $<userID_i>$ είναι το αναγνωριστικό του χρήστη που αντιστοιχεί στον i-οστό κόμβο της αλυσίδας αυτής.

- **U < userID >**

  Γεγονός τύπου *unregister user* το οποίο σηματοδοτεί τη διαγραφή ενός χρήστη (user) με αναγνωριστικό < userID > από τον πίνακα κατακερματισμού χρηστών. Πριν την οριστική διαγραφή του χρήστη από τη λίστα χρηστών θα πρέπει να διαγράψετε όλα τα στοιχεία του δένδρου ιστορικού ταινιών του χρήστη αν αυτό περιέχει στοιχεία.

  Κατά το γεγονός αυτό εντοπίζεται η κατάλληλη αλυσίδα βάσει της συνάρτησης κατακερματισμού και στη συνέχεια εκτελείται αναζήτηση ώστε να βρεθεί ο κατάλληλος κόμβος της αλυσίδας.

  Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
U <userID>
Chain <j> of Users:
    <userID1>
    <userID2>
       ...
    <userIDn>
DONE
```

όπου <j> είναι η τιμή κατακερματισμού του κλειδιού <userID>, n είναι ο αριθμός των χρηστών στην αλυσίδα που δεικτοδοτείται από τη θέση <j> του πίνακα Users και για κάθε i ∈ {1, …, n}, <userID$_i$> είναι το αναγνωριστικό του χρήστη που αντιστοιχεί στον i-οστό κόμβο της αλυσίδας αυτής.

– **A <movieID > <category> <year>**

Γεγονός τύπου *add new movie,* το οποίο σηματοδοτεί την άφιξη μιας νέας ταινίας που είναι διαθέσιμη στους χρήστες. Κατά το γεγονός αυτό, θα δημιουργείται μια νέα ταινία με αναγνωριστικό <movieID > και έτος κυκλοφορίας <year>, η οποία θα ανήκει στη θεματική κατηγορία <category>. **Ανεξάρτητα από την κατηγορία στην οποία ανήκει, η νέα ταινία θα εισάγεται στο δένδρο νέων κυκλοφοριών.** Τα πεδία watchedCounter και sumScore θα αρχικοποιούνται με την τιμή 0.

Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
A <movieID> <category> <year>
New releases Tree:
   <new_releases>: <movieID1>, ... , < movieIDn>
DONE
```

όπου n είναι το μέγεθος του δένδρου νέων κυκλοφοριών. Οι κόμβοι θα πρέπει να έχουν εισαχθεί με τέτοιο τρόπο ώστε αν πραγματοποιηθεί **ενδοδιατεταγμένη** (in-order) **διάσχιση στο δένδρο**, οι ταινίες να προσπελάζονται σε **αύξουσα διάταξη ως προς το πεδίο movieID.**

– **C**

Γεγονός τύπου *categorize movies* το οποίο σηματοδοτεί την ταξινόμηση των ταινιών που περιέχει το δένδρο νέων κυκλοφοριών στις υπόλοιπες θεματικές κατηγορίες. Σε αυτό το γεγονός, θα διασχίζετε το δένδρο των νέων κυκλοφοριών και για κάθε κόμβο, v, που βρίσκετε σ'αυτό θα εισάγετε έναν κόμβο στο δένδρο της κατάλληλης θεματικής κατηγορίας. Στη συνέχεια, θα διαγράφετε τον v από το δένδρο νέων κυκλοφοριών.

Για κάθε ταινία που προστίθεται στην κατηγορία i, θα πρέπει να αυξήσετε τον μετρητή κόμβων (movieCounter) της κατηγορίας.

Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
C
Movie Category Array:
    <category_0>: <movieID^0_1>, ... , < movieID^0_n0>
    <category_1>: <movieID^1_1>, ... , < movieID^1_n1>
    ...
   <category_4>: <movieID^4_1>, ... , < movieID^4_n4>
DONE
```

όπου για κάθε i, $0 \leq i \leq 4$, n_i είναι το μέγεθος του δένδρου ταινιών της κατηγορίας i του πίνακα κατηγοριών και για κάθε j, $1 \leq j \leq n_i$, $\texttt{<movieID}^{i}_{j}\texttt{>}$ είναι το αναγνωριστικό της ταινίας που αντιστοιχεί στον j-οστό κόμβο του δένδρου ταινιών της κατηγορίας i, όπως προκύπτει από την ενδοδιατεταγμένη διάσχισή του δένδρου αυτού.

– **G <userID > <movieID> <score>**

Γεγονός τύπου *rate movie* το οποίο σηματοδοτεί ότι ο χρήστης με αναγνωριστικό <userID> έχει παρακολουθήσει την ταινία με αναγνωριστικό <movieID> και την αξιολογεί με βαθμό <score>.

Κατά το γεγονός αυτό, θα γίνεται αναζήτηση της ταινίας με αναγνωριστικό <movieID> στα δένδρα κατηγοριών του πίνακα κατηγοριών. Όταν βρεθεί ο κόμβος της ταινίας, το πεδίο watchedCounter θα αυξάνεται κατά ένα και το πεδίο sumScore θα αυξάνεται κατά την τιμή <score>. Επιπρόσθετα, θα υπολογίσετε την τιμή του πεδίου medianScore του struct σύμφωνα με τον τύπο:

$$medianScore = \frac{sumScore}{watchedCounter}$$

Στη συνέχεια, θα δημιουργείτε έναν κόμβο, userMovie. Ο κόμβος αυτός θα έχει στα πεδία movieID και category ίδιες τιμές με εκείνες του struct που αντιστοιχεί στην ταινία με αναγνωριστικό movieID. Το πεδίο score του κόμβου userMovie θα έχει την τιμή <score>. Στη συνέχεια θα εισάγετε αυτόν τον κόμβο, στο φυλλο-προσανατολισμένο δένδρο ιστορικού του χρήστη με αναγνωριστικό <userID>.

Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
G <userID> <movieID> <score>
History Tree of user <userID>:
       <userMovieID1, score1>
       <userMovieID2, score2>
       …
       <userMovieIDn, scoren>
DONE
```

όπου n είναι το μέγεθος του δένδρου ιστορικού ταινιών του χρήστη με αναγνωριστικό <userID>, για κάθε i ∈ {1, …, n}, <movieIDi> είναι το αναγνωριστικό της ταινίας που αντιστοιχεί στον i-οστό κόμβο του δένδρου όπως προκύπτει από την ενδοδιατεταγμένη διάσχισή του και $<score_i>$ είναι η βαθμολογία της ταινίας που αντιστοιχεί στον κόμβο αυτό.

– **S <category>**

Γεγονός τύπου *cluster movies*. Στο γεγονός αυτό, θα πρέπει να ταξινομείτε τις ταινίες της κατηγορίας <category> με βάση το μέσο score των ταινιών, medianScore.

Για το σκοπό αυτό θα πρέπει να χρησιμοποιήσετε ένα βοηθητικό πίνακα μεγέθους ίσο με τον αριθμό των ταινιών που περιέχει η κατηγορία <category> (η τιμή αυτή είναι ήδη αποθηκευμένη στο struct της θέσης <category> του πίνακα κατηγοριών). Το κάθε στοιχείο του βοηθητικού πίνακα θα περιέχει έναν δείκτη σε ένα struct τύπου movie.

Στη συνέχεια θα εκτελείτε ενδοδιατεταγμένη διάσχιση του δένδρου ταινιών της κατηγορίας <category> και θα αποθηκεύεται δείκτες προς τα στοιχεία του δένδρου στο βοηθητικό πίνακα. Μετά το τέλος της διάσχισης του δένδρου (και την αρχικοποίηση του βοηθητικού πίνακα), θα εφαρμόζετε τον αλγόριθμο heapsort, για να ταξινομήσετε τον πίνακα βάσει του πεδίου medianScore. Τέλος θα εκτυπώνετε τις ταινίες σε 3 ομάδες: υψηλά βαθμολογημένες ταινίες (medianScore > 7), μέτρια βαθμολογημένες ταινίες (η τιμή του medianScore είναι μεταξύ 5 και 7), χαμηλά βαθμολογημένες ταινίες (medianScore < 5).

Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
S <category>
High Rated Movies [score>7]
       <movieID⁰₁> <medianScore⁰₁>
       …
       <movieID⁰ₖ₀> <medianScore⁰ₖ₀>
Medium Rated Movies [7>=score>=5]
       <movieID¹₁> <medianScore¹₁>
       …
       <movieID¹ₖ₁> <medianScore¹ₖ₁>
Low Rated Movies [score<5]
       <movieID²₁> <medianScore²₁>
       …
       <movieID²ₖ₂> <medianScore²ₖ₂>
DONE
```

όπου για κάθε i, $0 \leq i \leq 2$, k_i είναι το πλήθος των ταινίων που έχουν βαθμολογηθεί σε κάθε διάστημα και για κάθε j, $1 \leq j \leq n_i$, <movieID^i_j> είναι το αναγνωριστικό της ταινίας που αντιστοιχεί στη j-οστή θέση του βοηθητικού πίνακα μετά την εφαρμογή του αλγόριθμου heapsort.

– **Q <userID>**

Γεγονός τύπου *users median rate* το οποίο σηματοδοτεί τον υπολογισμό και την εκτύπωση στατιστικών για τη βαθμολογία του χρήστη με αναγνωριστικό <userID>. Πιο συγκεκριμένα, σε αυτό το γεγονός θα πρέπει να ακολουθήσετε τα εξής βήματα:

Εντοπίζετε την κατάλληλη αλυσίδα χρήστη βάσει της συνάρτησης κατακερματισμού και εκτελείτε αναζήτηση ώστε να βρεθεί ο κατάλληλος κόμβος της αλυσίδας.

Στη συνέχεια, διασχίζετε το δένδρο ιστορικού και αποθηκεύετε σε μια βοηθητική μεταβλητή, ScoreSum, το άθροισμα του score των κόμβων του δένδρου. Επιπρόσθετε, αποθηκεύετε σε μια άλλη βοηθητική μεταβλητή, counter, το πλήθος των ταινιών που είναι αποθηκευμένες στο δένδρο. Τέλος, διαιρείτε το ScoreSum με τον counter για να βρείτε τη μέση βαθμολογία που έχει δώσει ο χρήστης με αναγνωριστικό <userID> στις ταινίες που έχει παρακολουθήσει.

Η διάσχιση των φύλλων του φυλλο-προσανατολισμένου δένδρου θα πρέπει να πραγματοποιείται ως εξής: Αρχικά, θα βρίσκετε το αριστερότερο φύλλο, v, στο δένδρο. Για να βρείτε το φύλλο με το αμέσως μεγαλύτερο κλειδί από εκείνο του v, θα πρέπει να υλοποιήσετε αλγόριθμο, ο οποίος θα παίρνει ως παράμετρο έναν δείκτη στον κόμβο v και θα εκτελείται σε χρόνο O(h).

Να σημειωθεί εδώ ότι ο μέσος όρος που εσείς υπολογίζετε εδώ αναφέρεται σε συγκεκριμένο χρήστη και είναι διαφορετικός από το medianScore που αναφέρεται στο struct movie.

Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
Q <userID> <MScore>
DONE
```

όπου MScore, είναι ο μέσος όρος βαθμολογίας στις ταινίες που έχει παρακολουθήσει ο χρήστης που υπολογίζεται όπως περιγράφεται παραπάνω.

– **I <movieID> <category>**

Γεγονός τύπου *search movie* το οποίο σηματοδοτεί την αναζήτηση της ταινίας με αναγνωριστικό <movieID> στο δένδρο ταινιών της κατηγορίας <category>.

Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
I <movieID> <category> <year> <medianScore>
DONE
```

όπου <year> είναι το έτος κυκλοφορίας της ταινίας με αναγνωριστικό <movieID> και <medianScore> είναι ο μέσος όρος βαθμολογίας.

– **M**

Γεγονός τύπου *print movies* το οποίο σηματοδοτεί την εκτύπωση του δένδρου ταινιών όλων των κατηγοριών. Σε αυτό το γεγονός θα πρέπει για κάθε κατηγορία να εκτελέσετε ενδοδιατεταγμένη διάσχιση στο δένδρο και να τυπώσετε τους κόμβους που διασχίζετε.

Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
M
Movie Category Array:
    <category0>: <movieID01>, ... , < movieID0n0>
    <category1>: <movieID11>, ... , < movieID1n1>
    ...
   <category4>: <movieID41>, ... , < movieID4n4>
DONE
```

όπου για κάθε i, $0 \leq i \leq 4$, n_i είναι το μέγεθος του δένδρου ταινιών της κατηγορίας i του πίνακα κατηγοριών και για κάθε j, $1 \leq j \leq n_i$, <movieID^i_j> είναι το αναγνωριστικό της ταινίας που αντιστοιχεί στον j-οστό κόμβο του δένδρου ταινιών της κατηγορίας i, όπως προκύπτει από την ενδοδιατεταγμένη διάσχισή του δένδρου αυτού.

– **P**

Γεγονός τύπου *print users* το οποίο σηματοδοτεί την εκτύπωση του πίνακα κατακερματισμού χρηστών. Για τον κάθε χρήστη θα πρέπει να εκτυπώνονται όλα τα πεδία του struct που του αντιστοιχεί (εκτός από τους δείκτες), συμπεριλαμβανομένου του δένδρου ιστορικού.

Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
P

…

Chain <j> of Users:
       <userID1>
        History Tree:
              <movieID11> <score11>
              …
              <movieID1n1> <score1n1>
       <userID2>
        History Tree:
              <movieID21> <score21>
              …
              <movieID2n2> <score2n2>
       …
       <userIDn>
        History Tree:
              <movieIDn1> <scoren1>
              …
              <movieIDnnm> <scorennm>
…
DONE
```

όπου <j> είναι η j-στη αλυσίδα του πίνακα κατακερματισμού και n είναι ο αριθμός των χρηστών στην αλυσίδα που δεικτοδοτείται από τη θέση <j> του πίνακα Users και για κάθε $i \in \{1, \ldots, n\}$, $<userID_i>$ είναι το αναγνωριστικό του χρήστη που αντιστοιχεί στον i-οστό κόμβο της αλυσίδας αυτής. Επίσης, , $1 \leq k \leq n_i$, $<movieID^i_k>$ είναι η ταίνια από το δένδρο ιστορικού του χρήστη με αναγνωριστικό $<userID_i>$. Σε αυτό το γεγονός θα πρέπει να τυπώνεται ολόκληρος ο πίνακας κατακερματισμού.

– **W**

Γεγονός τύπου *world print* το οποίο σηματοδοτεί την εκτύπωση όλων των δομών δεδομένων που υπάρχουν στο σύστημα. Οι κόμβοι των δένδρων που υπάρχουν στο σύστημα θα πρέπει να διασχιστούν **με ενδοδιατεταγμένη διάταξη (in-order)**, δηλαδή θα πρέπει να εξασφαλίσετε ότι οι εισαγωγές είναι τέτοιες, έτσι ώστε με ενδοδιατεταγμένη διάταξη οι κόμβοι να τυπώνονται σε αύξουσα σειρά.

Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
W

…

Chain <j> of Users:

       <userID1>
        History Tree: <movieID1 1>, ... , < movieID0 m1>

       <userID2>
        History Tree: <movieID2 1>, ... , < movieID0 m2>

       …

       <userIDn>
        History Tree: <movieIDn 1>, ... , < movieID0 mn>

…

Movie Category Array:

    <category0>: <movieID0 1>, ... , < movieID0 n0>

    <category1>: <movieID1 1>, ... , < movieID1 n1>

    ...

   <category4>: <movieID4 1>, ... , < movieID4 n4>

New releases Tree:

   <new_releases>: <movieID1>, ... , < movieIDn>

DONE
```

όπου <j> είναι η j-στη αλυσίδα του πίνακα κατακερματισμού και n είναι ο αριθμός των χρηστών στην αλυσίδα που δεικτοδοτείται από τη θέση <j> του πίνακα Users και για κάθε $i \in \{1, \ldots, n\}$, <userID$_i$> είναι το αναγνωριστικό του χρήστη που αντιστοιχεί στον i-οστό κόμβο της αλυσίδας αυτής. Επίσης, , $1 \leq k \leq m_i$, <movieID^i_k> είναι η ταίνια από το δένδρο ιστορικού του χρήστη με αναγνωριστικό <userID$_i$>. Σε αυτό το γεγονός θα πρέπει να τυπώνεται ολόκληρος ο πίνακας κατακερματισμού. Στην εκτύπωση του πίνακα κατηγοριών έχουμε ότι για κάθε h, $0 \leq h \leq 4$, n_h είναι το μέγεθος του δένδρου ταινιών της κατηγορίας h του πίνακα κατηγοριών και για κάθε j το αναγνωριστικό της ταινίας που αντιστοιχεί στον j-οστό κόμβο του δένδρου ταινιών της κατηγορίας h, όπως προκύπτει από την ενδοδιατεταγμένη διάσχισή του δένδρου αυτού. Επίσης, θα τυπώνετε το δένδρο νέων κυκλοφοριών όπου n είναι το μέγεθος του.

Δομές Δεδομένων

Στην υλοποίησή σας δεν επιτρέπεται να χρησιμοποιήσετε έτοιμες δομές δεδομένων (πχ., ArrayList) είτε η υλοποίηση πραγματοποιηθεί στη C, C++ είτε στη Java. Στη συνέχεια παρουσιάζονται οι δομές σε C που πρέπει να χρησιμοποιηθούν για την υλοποίηση της παρούσας εργασίας.

```
/**
* Structure defining a node of movie binary tree (dendro tainiwn kathgorias)
*/
typedef struct movie{
    int movieID;                /* The movie identifier*/
    int category;               /* The category of the movie*/
    int year;                   /* The year movie released*/
    int watchedCounter;         /* How many users rate the movie*/
    int sumScore;               /* The sum of the ratings of the movie*/
    float medianScore;          /* Median score of the ratings of the movie*/
    struct movie *leftChild;    /* Pointer to the node's left child*/
    struct movie *rightChild;   /* Pointer to the node's right child*/
}movie_t;


/**
* Structure defining movie_category
*/

typedef struct movie_category{
    movie_t *movie;             /* Pointer to movie tree */
    movie_t *sentinel;          /* Pointer to movie tree sentinel */
    int movieCounter;           /* The number of movies in the category i */
}movieCategory_t;

/**
* Structure defining a node of user_movie for history doubly linked binary
* tree (dentro istorikou)
*/
typedef struct user_movie{
    int movieID;                    /* The movie identifier*/
    int category;                   /* The category of the movie*/
    int score;                      /* The score of the movie*/
    struct user_movie *parent;      /* Pointer to the node's parent*/
    struct user_movie *leftChild;   /* Pointer to the node's left child*/
    struct user_movie *rightChild;  /* Pointer to the node's right child*/
}userMovie_t;

/**
* Structure defining a node of users' hashtable (pinakas katakermatismou
* xrhstwn)
*/
typedef struct user {
    int userID;             /* The user's identifier*/
    userMovie_t *history;   /* A doubly linked binary tree with the movies
watched by the user*/
    struct user *next;      /* Pointer to the next node of the chain*/
}user_t;

/* Global variables for simplicity. */
```

```
movieCategory_t *categoryArray[5];  /* The categories array (pinakas
kathgoriwn)*/
movie_t *new_releases;     /* New releases simply-linked binary tree*/
user_t **user_hashtable_p;  /* The users hashtable. This is an array of
chains (pinakas katakermatismoy xrhstwn)*/
int hashtable_size;         /* The size of the users hashtable)*/
int max_users;              /* The maximum number of registrations (users)*/
int max_id;                 /* The maximum acount ID */
int primes_g[160];          /* Prime numbers for hashing*/
```