HY240: Δομές Δεδομένων
Χειμερινό Εξάμηνο – Ακαδημαϊκό Έτος 2023-2024
Διδάσκουσα: Παναγιώτα Φατούρου
Προγραμματιστική Εργασία – 1η Φάση

Ημερομηνία Παράδοσης: Δευτέρα, 20 Νοεμβρίου 2023, 23:59
Τρόπος Παράδοσης: Μέσω του προγράμματος turnin. Πληροφορίες για την χρήση του turnin στην ιστοσελίδα του μαθήματος (https://www.csd.uoc.gr/~hy240/current/submit.php)

Γενική Περιγραφή

Στην εργασία αυτή καλείστε να υλοποιήσετε μία απλοποιημένη υπηρεσία παρακολούθησης ταινιών (streaming service). Η υπηρεσία διαθέτει ταινίες ταξινομημένες σε διαφορετικές θεματικές κατηγορίες. Χρήστες εγγράφονται στην υπηρεσία, παρακολουθούν ταινίες προσθέτοντάς τις στο ιστορικό τους, δέχονται προτάσεις ταινιών βάσει του ιστορικού παρακολούθησης των υπόλοιπων χρηστών και πραγματοποιούν φιλτραρισμένες αναζητήσεις σε κατηγορίες ταινιών.

Αναλυτική Περιγραφή Ζητούμενης Υλοποίησης

Η υπηρεσία που θα υλοποιήσετε ταξινομεί τις ταινίες που διαθέτει σε **6** θεματικές κατηγορίες: Horror, Science-Fiction, Drama, Romance, Documentary, Comedy. Κάθε ταινία ανήκει σε **μία μόνο** κατηγορία και διαθέτει ένα **μοναδικό** αναγνωριστικό. Θα υλοποιήσετε την κατηγοριοποίηση των ταινιών μέσω ενός πίνακα 6 θέσεων, του **πίνακα κατηγοριών**. Σε κάθε θέση του πίνακα βρίσκεται ένας δείκτης (τύπου ***struct movie ****) ο οποίος δείχνει στο πρώτο στοιχείο της λίστας ταινιών που αντιστοιχούν στη συγκεκριμένη κατηγορία. Αυτή η λίστα είναι **απλά-συνδεδεμένη** και **ταξινομημένη** σε **αύξουσα διάταξη** βάσει του αναγνωριστικού ταινίας (mid). Ένας κόμβος αυτής της λίστας περιγράφει μία ταινία που ανήκει στην κατηγορία, μέσω μίας δομής (***struct movie***) με τα εξής πεδία:

- **info**: Βοηθητική δομή τύπου ***struct movie_info*** που περιγράφει τις διαθέσιμες πληροφορίες για μία ταινία. Τα πεδία της είναι ως εξής:
    - **mid**: Μοναδικό αναγνωριστικό της ταινίας, τύπου ***unsigned int***.
    - **year**: Έτος κυκλοφορίας της ταινίας, τύπου ***unsigned int***.
- **next**: Δείκτης (τύπου ***struct movie***) που δείχνει στο επόμενο στοιχείο της λίστας ταινιών της κατηγορίας.

Πριν να εισαχθούν στην κατάλληλη λίστα του πίνακα κατηγοριών, οι καινούριες ταινίες που προστίθενται στην υπηρεσία, εισάγονται σε μία ξεχωριστή λίστα, την **λίστα νέων κυκλοφοριών**. Αυτή η λίστα περιέχει ταινίες διαφορετικών κατηγοριών και είναι **απλά-συνδεδεμένη** και **ταξινομημένη** σε **αύξουσα διάταξη** βάσει του αναγνωριστικού ταινίας, όπως και οι λίστες του πίνακα κατηγοριών. Οι κόμβοι της υλοποιούνται με την δομή ***struct new_movie***, η οποία έχει τα εξής πεδία:

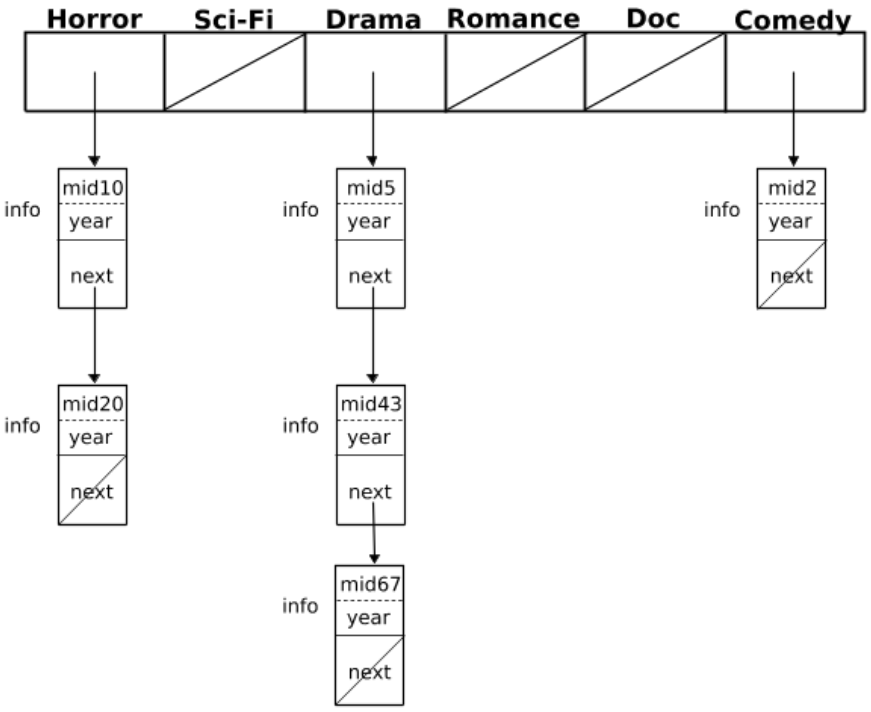
- **info**: Πληροφορίες σχετικά με την ταινία, τύπου ***struct movie_info***, όπως και στο struct movie.
- **category**: Η κατηγορία στην οποία ανήκει αυτή η ταινία, η οποία αναπαρίσταται ως ένα enum τύπου ***movieCategory_t***.
- **next**: Δείκτης (τύπου ***struct new_movie***) ο οποίος δείχνει στον επόμενο κόμβο της λίστας νέων κυκλοφοριών.

Στο Σχήμα 1 φαίνεται ο πίνακας κατηγοριών σε μία υποθετική εκτέλεση του προγράμματος. Είναι, σταθερού μεγέθους 6 θέσεων με μία λίστα ταινιών ανά θέση.

Οι χρήστες που είναι εγγεγραμμένοι στην υπηρεσία είναι οργανωμένοι σε μία **μη-ταξινομημένη**, **απλά-συνδεδεμένη** λίστα με **κόμβο φρουρό**, τη **λίστα χρηστών**. Τα στοιχεία της λίστας χρηστών είναι τύπου ***struct user***. Αυτή η δομή περιέχει για κάθε χρήστη, επιπροσθέτως του αναγνωριστικού του, έναν δείκτη στην αρχή μία διπλά-συνδεδεμένης λίστας, που ονομάζεται λίστα προτεινόμενων ταινιών του χρήστη και περιέχει τις ταινίες που προτείνει η υπηρεσία

στον χρήστη. Επίσης, η λίστα χρηστών περιέχει, για κάθε χρήστη, έναν δείκτη στο κορυφαίο στοιχείο μίας στοίβας, η οποία υλοποιεί το ιστορικό παρακολούθησης ταινιών του χρήστη. Τα πεδία του ***struct user*** είναι ως εξής:

- **uid**: Μοναδικό αναγνωριστικό του χρήστη, τύπου ***int***.
- **suggestedHead**: Δείκτης τύπου ***struct suggested_movie*** (περισσότερες πληροφορίες παρακάτω) ο οποίος δείχνει στο πρώτο στοιχείο της διπλά συνδεδεμένης λίστας προτεινόμενων ταινιών του χρήστη.
- **suggestedTail**: Δείκτης τύπου ***struct suggested_movie*** ο οποίος δείχνει στο τελευταίο στοιχείο της διπλά συνδεδεμένης λίστας προτεινόμενων ταινιών του χρήστη.
- **watchHistory**: Δείκτης τύπου ***struct movie*** (όπως ορίστηκε παραπάνω) ο οποίος δείχνει στην κορυφής της στοίβας ιστορικού παρακολούθησης ταινιών του χρήστη.
- **next**: Δείκτης τύπου ***struct user*** που δείχνει στον επόμενο κόμβο της λίστας χρηστών.

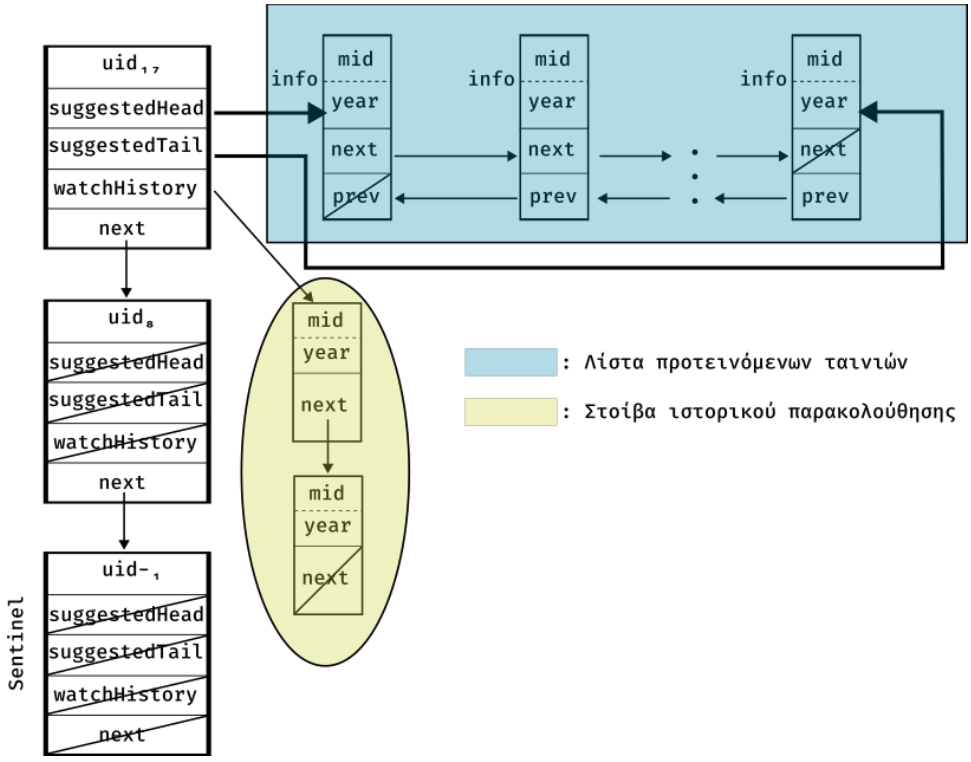


Σχήμα 1: Πίνακας κατηγοριών και λίστα ταινιών ανά κατηγορία. Κάθε λίστα είναι ταξινομημένη σε αύξουσα διάταξη βάσει αναγνωριστικού ταινίας (mid)

Ο κόμβος φρουρός της λίστας χρηστών είναι ένας κόμβος που χρησιμοποιείται για την ορθή διαχείριση της λίστας (και επομένως είναι βοηθητικός κόμβος). Είναι τύπου ***struct user***, αλλά το **uid** του ισούται με -1. Οι δείκτες **suggestedHead**, **suggestedTail**, **watchHistory** και **next** κάθε χρήστη έχουν αρχική τιμή ίση με NULL. Η πιθανή μορφή της λίστας χρηστών σε μία υποθετική εκτέλεση του προγράμματος απεικονίζεται στο Σχήμα 2.

Για την υλοποίηση της διπλά-συνδεδεμένης λίστας προτεινόμενων ταινιών του χρήστη θα χρησιμοποιήσετε την δομή struct suggested_movie, τα πεδία της οποίας είναι:

- **info**: Πεδίο τύπου ***struct movie_info***, το οποίο περιέχει τις πληροφορίες σχετικά με την ταινία.
- **prev**: Δείκτης τύπου ***struct suggested_movie***, ο οποίος δείχνει στο προηγούμενο στοιχείο της λίστας προτεινόμενων ταινιών.
- **next**: Δείκτης τύπου ***struct suggested_movie***, ο οποίος δείχνει στο επόμενο στοιχείο της λίστας προτεινόμενων ταινιών.



Σχήμα 2: Οργάνωση λίστας χρηστών

Τρόπος λειτουργίας προγράμματος

Το πρόγραμμα που θα υλοποιήσετε θα πρέπει να καλείται με την ακόλουθη εντολή:

<executable> <input file>

όπου <executable> είναι το όνομα του εκτελέσιμου σας (π.χ., cs240StreamingService) και <input file> είναι το όνομα ενός αρχείου εισόδου (π.χ., testFile), ένα αρχείο το οποίο περιέχει ένα γεγονός ανά γραμμή. Τα γεγονότα πρέπει να υλοποιηθούν από το πρόγραμμά σας και ακολουθούν μία από τις παρακάτω μορφές:

– R <uid>

Γεγονός τύπου *Register User*, στο οποίο ένας καινούριος χρήστης με αναγνωριστικό <uid> εγγράφεται στην υπηρεσία. Θα πρέπει αρχικά να διατρέξετε την λίστα χρηστών ώστε να ελέγξετε ότι δεν υπάρχει ήδη χρήστης με το αναγνωριστικό <uid> στην υπηρεσία. Εφόσον αυτό ισχύει θα πρέπει να υλοποιήσετε αλγόριθμο εισαγωγής του νέου χρήστη στη λίστα χρηστών. Ο αλγόριθμός σας θα πρέπει να έχει **χρονική πολυπλοκότητα O(1)**. Τα πεδία suggestedHead, suggestedTail και watchHistory του καινούριου χρήστη αρχικοποιούνται σε NULL. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
R <uid>
   Users = <uid_1>, <uid_2>, ... , <uid_n>
DONE
```

όπου n είναι ο αριθμός των κόμβων της λίστας χρηστών και για κάθε i ∈ {1, …, n} <uidi> είναι το αναγνωριστικό του χρήστη που αντιστοιχεί στον i-οστό κόμβο της λίστας.

– U <uid>

Γεγονός τύπου *Unregister User*, στο οποίο ο χρήστης με αναγνωριστικό <uid> αποχωρεί από την υπηρεσία. Θα πρέπει αρχικά να αδειάσετε την διπλά συνδεδεμένη λίστα προτεινόμενων ταινιών του χρήστη και την στοίβα ιστορικού παρακολούθησής του αφαιρώντας όλα τα στοιχεία τους -εφόσον υπάρχουν-, προτού αφαιρέσετε τον χρήστη από την

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

```
U <uid>
   Users = <uid_1>, <uid_2>, ... , <uid_n>
DONE
```

όπου n είναι ο αριθμός των κόμβων της λίστας χρηστών και για κάθε i ∈ {1, …, n} <uidi> είναι το αναγνωριστικό του χρήστη που αντιστοιχεί στον i-οστό κόμβο της λίστας.

– A <mid> <category> <year>

Γεγονός τύπου *Add New Movie*, στο οποίο η ταινία με αναγνωριστικό <mid>, κατηγορία <category> και έτος κυκλοφορίας <year> προστίθεται στην υπηρεσία. **Θα πρέπει να εισάγετε την ταινία στην ταξινομημένη λίστα νέων κυκλοφοριών, και όχι στην λίστα του πίνακα κατηγοριών.** Η λίστα νέων κυκλοφοριών θα πρέπει να παραμένει ταξινομημένη (σε αύξουσα διάταξη βάσει του αναγνωριστικού ταινίας) μετά από κάθε εισαγωγή. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
A <mid> <category> <year>
   New movies = <mid_1,category_1,year_1>, <mid_2,category_2,year_2>, ... , <mid_n, category_n,year_n>
DONE
```

όπου n είναι ο αριθμός στοιχείων της λίστας νέων κυκλοφοριών και mid_i, category_i, year_i, i ∈ {1, …, n}, είναι αντίστοιχα το αναγνωριστικό, η κατηγορία και το έτος κυκλοφορίας της ταινίας, που αντιστοιχούν στον i-οστό κόμβο της λίστας νέων κυκλοφοριών.

– D

Γεγονός τύπου *Distribute New Movies*, στο οποίο οι ταινίες της λίστας νέων κυκλοφοριών ανατίθενται, βάσει της κατηγορίας στην οποία ανήκουν, στις λίστες του πίνακα κατηγοριών. Το γεγονός αυτό θα πρέπει να υλοποιείται σε **χρονική πολυπλοκότητα O(n)**, όπου n είναι το μέγεθος της λίστας νέων κυκλοφοριών. Θα χρειαστεί να διασχίσετε μία φορά την λίστα νέων κυκλοφοριών αφαιρώντας όποια ταινία συναντάτε και προσθέτοντάς την στην κατάλληλη λίστα του πίνακα κατηγοριών. Στο τέλος αυτής της διαδικασίας η λίστα νέων κυκλοφοριών θα πρέπει να έχει αδειάσει και οι λίστες του πίνακα κατηγοριών να είναι ταξινομημένες σε αύξουσα διάταξη βάσει του αναγνωριστικού ταινίας. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
D
Categorized Movies:
   Horror: <mid_1,1>, <mid_1,2>, ... , <mid_1,n1>
   Sci-fi: <mid_2,1>, <mid_2,2>, ... , <mid_2,n2>
   Drama: <mid_3,1>, <mid_3,2>, ... , <mid_3,n3>
   Romance: <mid_4,1>, <mid_4,2>, ... , <mid_4,n4>
   Documentary: <mid_5,1>, <mid_5,2>, ... , <mid_5,n5>
   Comedy: <mid_6,1>, <mid_6,2>, ... , <mid_6,n6>
DONE
```

όπου n1, n2, …, n6 είναι τα μεγέθη των 6 λιστών κατηγοριών και <mid_i,j>, είναι το αναγνωριστικό της ταινίας που αντιστοιχεί στον j-οστό κόμβο της λίστας ταινιών της κατηγορίας i.

– W <uid> <mid>

Γεγονός τύπου *User Watches Movie*, στο οποίο ο χρήστης με αναγνωριστικό <uid> παρακολουθεί την ταινία με αναγνωριστικό <mid>. Αρχικά, θα πρέπει να βρεθεί η πληροφορία της ταινίας (***struct movie_info***) αναζητώντας τις λίστες του πίνακα κατηγοριών αλλά και ο χρήστης, αναζητώντας την λίστα χρηστών. Στη συνέχεια δημιουργείτε ένα καινούριο ***struct movie*** και το σπρώχνετε στην στοίβα ιστορικού παρακολούθησης του χρήστη. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
W <uid> <mid>
   User <uid> Watch History = <mid_1>, <mid_2>, ... , <mid_n>
DONE
```

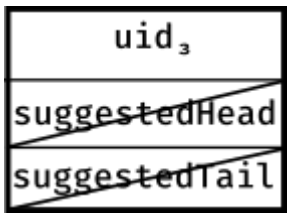
όπου n είναι ο αριθμός στοιχείων της στοίβας ιστορικού παρακολούθησης του χρήστη <uid> και mid_i, i ∈ {1, …, n}, το αναγνωριστικό ταινίας που αντιστοιχεί στο i-οστό στοιχείο της στοίβας.

– S <uid>

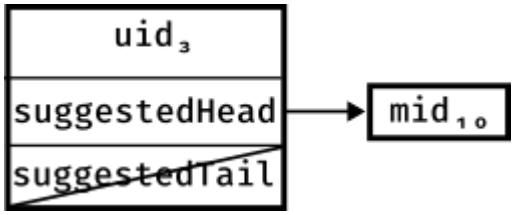
Γεγονός τύπου *Suggest Movies to User*, στο οποίο ο χρήστης με αναγνωριστικό <uid> δέχεται προτάσεις ταινιών από την υπηρεσία, βάσει του ιστορικού παρακολούθησης των υπόλοιπων χρηστών. Για κάθε χρήστη, στην λίστα χρηστών -εκτός του χρήστη <uid>- θα πρέπει να αφαιρέσετε μία ταινία από την κορυφή της στοίβας ιστορικού παρακολούθησής του -εφόσον υπάρχει- και να την προσθέσετε στην διπλά συνδεδεμένη λίστα προτεινόμενων ταινιών του χρήστη <uid>. Οι προσθήκες αυτές θα πρέπει να γίνουν "**εναλλάξ**", στην αρχή και στο τέλος, όπως περιγράφεται στη συνέχεια. Το πρώτο στοιχείο θα εισαχθεί ως πρώτο στοιχείο και θα δείχνει σε αυτό ο δείκτης suggestedHead. Το δεύτερο στοιχείο θα εισαχθεί ως τελευταίο και θα δείχνει σ' αυτό ο δείκτης suggestedTail. Το (2i+1)-οστό στοιχείο, i > 0, θα μπει ως επόμενο του (2i-1)-οστού στοιχείου, ενώ το (2i)-οστό στοιχείο, i > 1, θα εισαχθεί ως προηγούμενο του (2i-2)-οστού στοιχείου. Επομένως, το τρίτο στοιχείο θα εισαχθεί ως επόμενο του πρώτου, ενώ το τέταρτο στοιχείο θα εισαχθεί ως προηγούμενο του δεύτερου, κ.ο.κ.

Για παράδειγμα, έστω ότι στη λίστα χρηστών υπάρχουν 5 χρήστες -πλην του κόμβου φρουρού-, με uids 7, 3, 15, 22, 37, και έστω ότι το ζητούμενο γεγονός είναι S 3. Στην προκειμένη περίπτωση θέλουμε να αφαιρέσουμε μία ταινία από τις στοίβες ιστορικού παρακολούθησης των χρηστών 7, 15, 22, 37, και σε κάθε βήμα η λίστα προτεινόμενων ταινιών του χρήστη 3 θα διαμορφωθεί ως εξής:

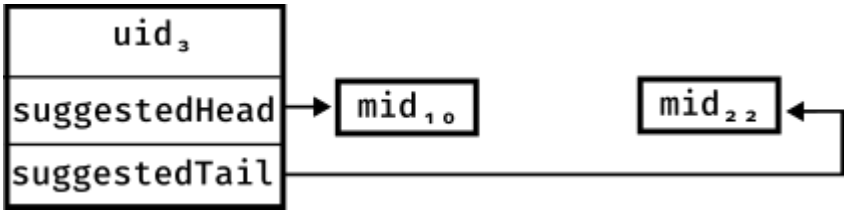
- Αρχική κατάσταση: η λίστα προτεινόμενων ταινιών του χρήστη 3 είναι κενή.



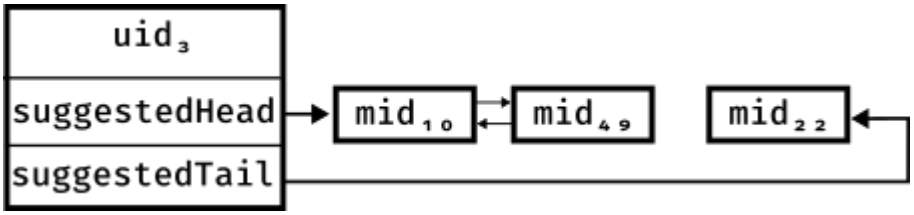
- Pop (User 7 Watch History) → mid 10



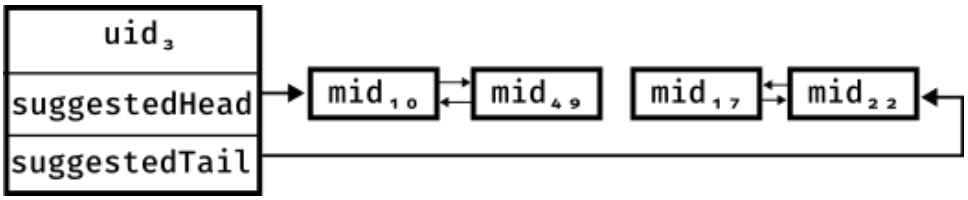
- Pop (User 15 Watch History) → mid 22



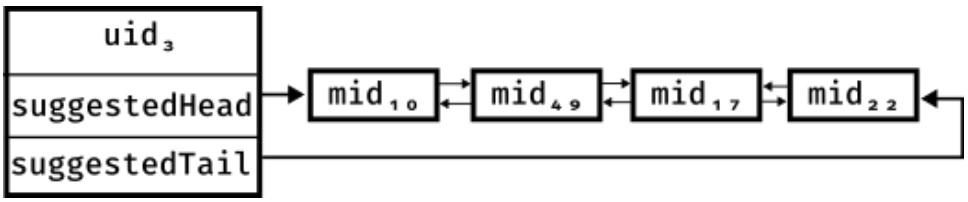
- Pop (User 22 Watch History) → mid 49



- Pop (User 37 Watch History) → mid 17



- Τελική λίστα προτεινόμενων ταινιών του χρήστη 3:



Η διαδικασία αυτή θα πρέπει να εκτελείται με χρονική πολυπλοκότητα O(n), όπου n είναι ο αριθμός χρηστών της εφαρμογής. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
S <uid>
   User <uid> Suggested Movies = <mid_1>, <mid_2>, ... , <mid_n>
DONE
```

όπου n είναι ο αριθμός στοιχείων στην λίστα προτεινόμενων ταινιών του χρήστη <uid> και mid_i, i ∈ {1, …, n}, είναι το αναγνωριστικό ταινίας που αντιστοιχεί στον i-οστό κόμβο της λίστας προτεινόμενων ταινιών.

– F <uid> <category1> <category2> <year>

Γεγονός τύπου *Filtered Movie Search*, στο οποίο ο χρήστης <uid> ζητάει από την υπηρεσία να του προτείνει ταινίες που να ανήκουν σε μία από τις δύο κατηγορίες <category1> και <category2> και να έχουν έτος κυκλοφορίας μεγαλύτερο ή ίσο του <year>. Θα πρέπει να διατρέξετε τις λίστες ταινιών για τις δύο κατηγορίες στον πίνακα κατηγοριών, να εντοπίσετε τις ταινίες με κατάλληλο έτος κυκλοφορίας και να τις ενώσετε σε μία νέα διπλά συνδεδεμένη λίστα. Αυτή η λίστα θα πρέπει να είναι **ταξινομημένη σε αύξουσα διάταξη βάσει του αναγνωριστικού ταινίας**, όπως και οι λίστες κατηγοριών, και να ενωθεί με το τέλος της υπάρχουσας λίστας προτεινόμενων ταινιών του χρήστη -εφόσον αυτή δεν είναι κενή, αλλιώς η λίστα προτεινόμενων ταινιών του χρήστη θα ισούται με τη νέα λίστα. **Δεν πρέπει να αφαιρέσετε στοιχεία από τις δύο λίστες κατηγοριών**, αντ’ αυτού θα πρέπει για κάθε ταινία που βρίσκετε να δημιουργείτε έναν καινούριο κόμβο της λίστας προτεινόμενων ταινιών του χρήστη.

Η διαδικασία θα πρέπει να εκτελείται σε **χρονική πολυπλοκότητα O(n + m), όπου n, m είναι αντίστοιχα ο αριθμός στοιχείων των δύο λιστών κατηγορίας που θα διατρέξετε**. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
F <uid> <category1> <category2> <year>
   User <uid> Suggested Movies = <mid_1>, <mid_2>, ... , <mid_n>
DONE
```

όπου n είναι ο αριθμός στοιχείων στην λίστα προτεινόμενων ταινιών του χρήστη <uid> και mid_i, i ∈ {1, …, n}, είναι το αναγνωριστικό ταινίας που αντιστοιχεί στον i-οστό κόμβο της λίστας προτεινόμενων ταινιών.

– T <mid>

Γεγονός τύπου *Take Off Movie*, όπου η ταινία <mid> αφαιρείται από την υπηρεσία. Θα πρέπει να αφαιρέσετε την ταινία από οποιαδήποτε λίστα προτεινόμενων ταινιών χρηστών μπορεί να βρίσκεται αλλά και από την κατάλληλη λίστα κατηγορίας του πίνακα κατηγοριών. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
T <mid>
   <mid> removed from <uid_1> suggested list.
   <mid> removed from <uid_2> suggested list.
   ...
   <mid> removed from <uid_n> suggested list.
   <mid removed from <category> category list.
   Category list = <mid_1>, <mid_k>
DONE
```

όπου n ο αριθμός των χρηστών οι οποίοι είχαν την ταινία <mid> στην λίστα προτεινόμενων ταινιών τους, uid_i, i ∈ {1, …, n} είναι το αναγνωριστικό του i-οστού χρήστη στη λίστα χρηστών που είχε την ταινία στην λίστα προτεινόμενων του, <category> η κατηγορία στην οποία άνηκε η ταινία, k το νέο μέγεθος της λίστας κατηγορίας στην οποία άνηκε η <mid> μετά την αφαίρεσή της και mid_j, j ∈ {1, …, k} το αναγνωριστικό της j-οστής ταινίας στην λίστα κατηγορίας. *Συμβουλή: για να μην χρειάζεται να "θυμάστε" από ποιες λίστες προτεινόμενων αφαιρέσατε την ταινία μπορείτε ανά μία αφαίρεση που κάνετε να τυπώνετε και μία γραμμή πληροφορίας.*

– M

Γεγονός τύπου *Print Movies* στο οποίο θα πρέπει να εκτυπώσετε πληροφορίες για όλες τις ταινίες στις λίστες του πίνακα κατηγοριών. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
M
Categorized Movies:
   Horror: <mid_1,1>, <mid_1,2>, ... , <mid_1,n1>
   Sci-fi: <mid_2,1>, <mid_2,2>, ... , <mid_2,n2>
   Drama: <mid_3,1>, <mid_3,2>, ... , <mid_3,n3>
   Romance: <mid_4,1>, <mid_4,2>, ... , <mid_4,n4>
   Documentary: <mid_5,1>, <mid_5,2>, ... , <mid_5,n5>
   Comedy: <mid_6,1>, <mid_6,2>, ... , <mid_6,n6>
DONE
```

όπου n1, n2, …, n6 είναι τα μεγέθη των 6 λιστών κατηγοριών και <mid_i,j>, είναι το αναγνωριστικό της ταινίας που αντιστοιχεί στον j-οστό κόμβο της i-οστής λίστας κατηγορίας.

– P

Γεγονός τύπου *Print Users* στο οποίο θα πρέπει να εκτυπώσετε πληροφορίες για κάθε χρήστη στη λίστα χρηστών. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
P
Users:
   <uid_1>:
      Suggested: <mid_1,1,>, <mid_1,2>, ... , <mid_1,s_1>
      Watch History: <mid'_1,1,>, <mid'_1,2>, ... , <mid'_1,w_1>
   <uid_2>:
      Suggested: <mid_2,1>, <mid_2,2>, ... , <mid_2,s_2>
      Watch History: <mid'_2,1>, <mid'_2,2>, ... , <mid'_2,w_2>
   ....
   <uid_n>
      Suggested: <mid_n,1>, <mid_n,2>, ... , <mid_n, s_n>
      Watch History: <mid'_n,1>, <mid'_n,2>, ... , <mid'_n,w_n>
DONE
```

όπου n είναι ο αριθμός στοιχείων της λίστας χρηστών, s_i, w_i, i ∈ {1, …, n} είναι αντίστοιχα το μέγεθος της λίστας προτεινόμενων ταινιών και της στοίβας ιστορικού παρακολούθησης του i-οστού χρήστη, mid_i,j, i ∈ {1, …, n}, j∈{1,...,s_i} είναι το αναγνωριστικό ταινίας του j-οστού κόμβου της λίστας προτεινόμενων ταινιών του i-οστού χρήστη και mid'_i,j, i ∈ {1, …, n}, j∈{1,...,w_i} είναι το αναγνωριστικό ταινίας του j-οστού κόμβου της στοίβας ιστορικού παρακολούθησης του i-οστού χρήστη.

Δομές Δεδομένων

Στην υλοποίησή σας δεν επιτρέπεται να χρησιμοποιήσετε έτοιμες δομές δεδομένων (π.χ., ArrayList) ανεξαρτήτως της γλώσσας προγραμματισμού που θα επιλέξετε (C, C++, Java). Στη συνέχεια παρουσιάζονται οι δομές σε C που χρειάζονται να υλοποιηθούν για την εργασία:

```
typedef enum {
        HORROR,
        SCIFI,
        DRAMA,
        ROMANCE,
        DOCUMENTARY,
        COMEDY
} movieCategory_t;

struct movie_info {
        unsigned mid;
        unsigned year;
};

struct movie {
        struct movie_info info;
        struct movie *next;
};

struct new_movie {
        struct movie_info info;
        movieCategory_t category;
        struct new_movie *next;
};

struct suggested_movie {
        struct movie_info info;
        struct suggested_movie *prev;
        struct suggested_movie *next;
};

struct user {
        int uid;
        struct suggested_movie *suggestedHead;
        struct suggested_movie *suggestedTail;
        struct movie *watchHistory;
        struct user *next;
};
```