HY240: Δομές Δεδομένων
Εαρινό Εξάμηνο – Ακαδημαϊκό Έτος 2016
Διδάσκουσα: Παναγιώτα Φατούρου
Προγραμματιστική Εργασία - 1ο Μέρος

Ημερομηνία Παράδοσης: Κυριακή, 3 Απριλίου 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, να εντοπίζουν τις αγαπημένες τους ταινίες, καθώς επίσης να βρίσκουν προτεινόμενες ταινίες με βάση τις αγαπημένες τους.

Αναλυτική Περιγραφή Ζητούμενης Υλοποίησης

Η υπηρεσία διαθέτει μια συλλογή ταινιών ταξινομημένες σε κατηγορίες. Πιο συγκεκριμένα, στα πλαίσια αυτής της προγραμματιστικής εργασίας, οι διαθέσιμες κατηγορίες ταινιών είναι οι παρακάτω: δραματικές, oscar, cinephile, ντοκιμαντέρ και κινούμενα σχέδια. Επιπρόσθετα, υπάρχει μια επιπλέον ξεχωριστή κατηγορία, «νέες κυκλοφορίες», η οποία περιλαμβάνει πρόσφατες ταινίες απ' όλες τις παραπάνω κατηγορίες.

Για τη διαχείριση αυτών των κατηγοριών, θα δημιουργήσετε έναν πίνακα σταθερού μεγέθους **6 θέσεων**, ο οποίος θα ονομάζεται **πίνακας κατηγοριών**. Κάθε θέση του πίνακα θα περιέχει έναν δείκτη (τύπου movie) στον πρώτο κόμβο μιας **απλά συνδεδεμένης λίστας**, η οποία θα ονομάζεται **λίστα ταινιών της κατηγορίας** αυτής. Ο κάθε κόμβος αυτής της λίστας περιέχει πληροφορίες για κάποια ταινία που ανήκει σε αυτήν την κατηγορία και θα αποτελεί μια εγγραφή (struct) τύπου movie με τα ακόλουθα πεδία:

- **mid**: Αναγνωριστικό (τύπου int) που χαρακτηρίζει μοναδικά την ταινία.
- **category**: Αναγνωριστικό (τύπου int) που αντιστοιχεί στη θεματική κατηγορία της ταινίας. Η μεταβλητή αυτή λαμβάνει τιμές από 0 μέχρι 4, με κάθε τιμή να αντιστοιχεί σε κάθε θεματική κατηγορία (0: drama, 1: Oscar, 2: cinephile, 3: documentary, 4: cartoon)
- **year**: Αριθμός (τύπου int) που αντιστοιχεί στο έτος κυκλοφορίας της ταινίας.
- **next**: Δείκτης (τύπου movie) στον επόμενο κόμβο της **λίστας ταινιών της κατηγορίας.**

Η λίστα ταινιών κάθε κατηγορίας είναι **ταξινομημένη,** σε αύξουσα διάταξη, βάσει των αναγνωριστικών των ταινιών.

Στο Σχήμα 1 παρουσιάζεται ο πίνακας κατηγοριών, σταθερού μεγέθους 6 θέσεων, και η λίστα των ταινιών που δεικτοδοτείται από κάθε στοιχείο του.

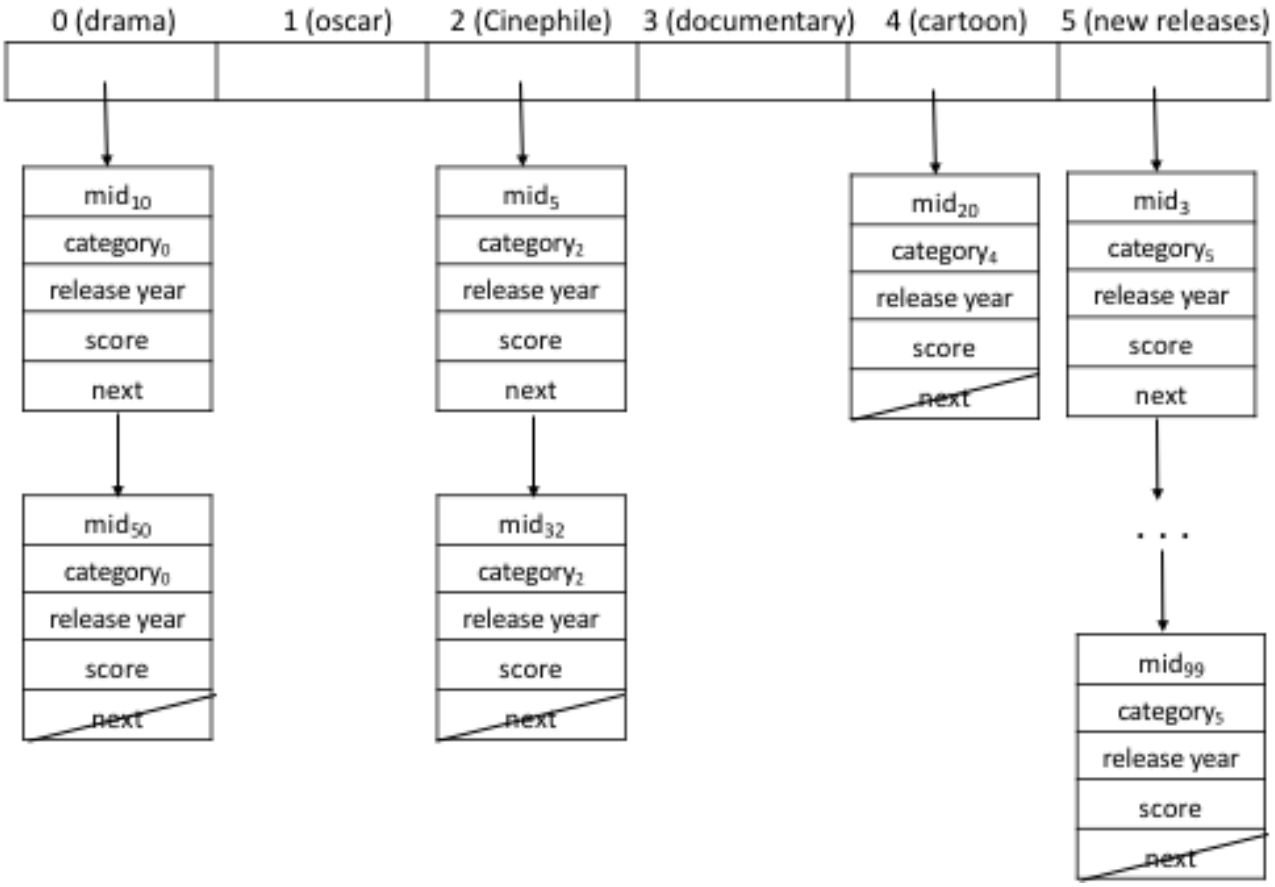
Η υπηρεσία εξυπηρετεί ένα σύνολο εγγεγραμμένων χρηστών. Για το σκοπό αυτό θα δημιουργήσετε μια **μη-ταξινομημένη, απλά συνδεδεμένη** λίστα **με κόμβο φρουρό,** η οποία θα ονομάζεται **λίστα χρηστών**. Κάθε κόμβος της λίστας χρηστών θα περιέχει μια εγγραφή (struct) τύπου user με τα ακόλουθα πεδία:

- **uid:** Αναγνωριστικό (τύπου int) που χαρακτηρίζει μοναδικά το χρήστη.
- **history:** Δείκτης (τύπου user_movie, βλέπε παρακάτω) στο πρώτο στοιχείο μιας **διπλά συνδεδεμένης** λίστας, η οποία ονομάζεται **λίστα ιστορικού ταινιών** του χρήστη**.** Η λίστα αυτή περιέχει ταινίες που έχει ήδη παρακολουθήσει και βαθμολογήσει ο χρήστης και είναι **ταξινομημένη,** σε φθίνουσα διάταξη, βάσει του score των ταινιών που περιέχει.
- **favorites**: Δείκτης (τύπου user_movie) στο πρώτο στοιχείο μιας **διπλά συνδεδεμένης** λίστας, η οποία ονομάζεται **λίστα αγαπημένων ταινιών** του χρήστη**.** Η λίστα αυτή περιέχει τις ταινίες που έχει ήδη παρακολουθήσει ο χρήστης και τις έχει βαθμολογήσει με score μεγαλύτερο ή ίσο από 7. Η λίστα αυτή είναι **ταξινομημένη,** σε φθίνουσα διάταξη, **βάσει του score** των ταινιών που περιέχει.
- **next**: Δείκτης (τύπου user) στον επόμενο κόμβο της λίστας.

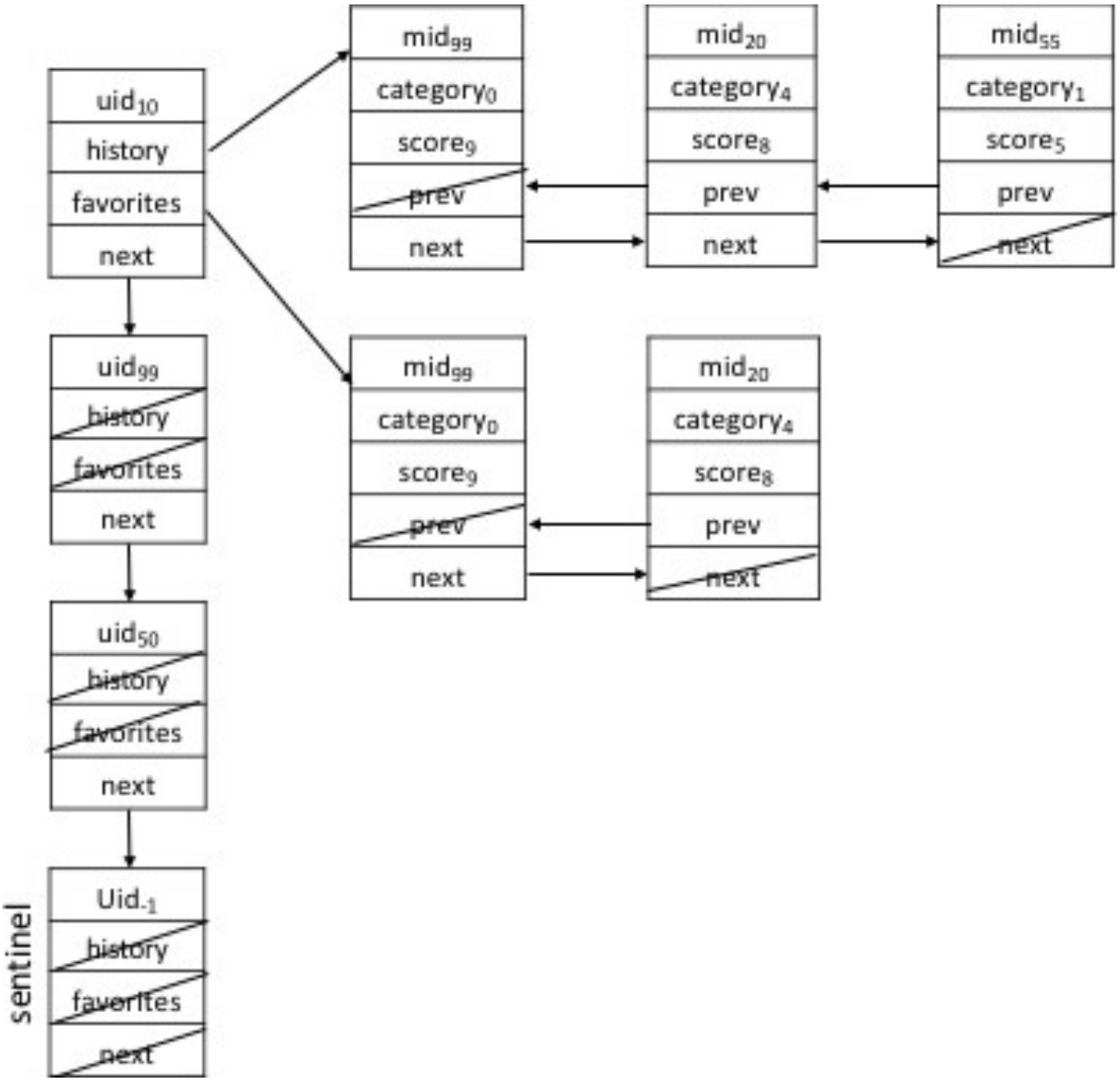
Ο κόμβος φρουρός της λίστας είναι ένας διαχειριστικός κόμβος για τον οποίο ισχύουν τα εξής: είναι και αυτός τύπου user με την ιδιαιτερότητα ότι το αναγνωριστικό του (mid) έχει τιμή -1 και τα πεδία δεικτών (history, favorites, next) έχουν τιμή NULL. Το Σχήμα 2 απεικονίζει τη λίστα χρηστών.

Ο κάθε κόμβος της λίστας ιστορικού και της λίστας αγαπημένων ταινιών του κάθε χρήστη αντιστοιχεί σε μια εγγραφή τύπου user_movie με τα ακόλουθα πεδία:

- **mid**: Αναγνωριστικό (τύπου int) που χαρακτηρίζει μοναδικά την ταινία.
- **category**: Αναγνωριστικό (τύπου int) που αντιστοιχεί στη θεματική κατηγορία της ταινίας. Η μεταβλητή αυτή λαμβάνει τιμές από 0 μέχρι 4, με κάθε τιμή να αντιστοιχεί σε κάθε θεματική κατηγορία (0: drama, 1: Oscar, 2: cinephile, 3: documentary, 4: cartoon)
- **score**: Αριθμός (τύπου int) που αντιπροσωπεύει τη βαθμολογία που δίνει ο χρήσης στην ταινία. Η μεταβλητή αυτή θα λαμβάνει τιμές απο 1 μέχρι 10, με 1 να αντιστοιχεί στον ελάχιστο βαθμό ικανοποίησης του χρήστη και 10 στο μέγιστο βαθμό ικανοποίησης.
- **prev**: Δείκτης (τύπου user_movie) στον προηγούμενο κόμβο της λίστας**.**
- **next**: Δείκτης (τύπου user_movie) στον επόμενο κόμβο της λίστας**.**



Σχήμα 1: Πίνακας κατηγοριών και αντίστοιχες λίστες ταινιών της κάθε κατηγορίας (κάθε μια από τις λίστες ταινιών είναι απλά συνδεδεμένη και ταξινομημένη βάσει του mid).



Σχήμα 2: Λίστα χρηστών

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 γεγονότα των ακόλουθων μορφών:

– **R <uid>**

Γεγονός τύπου *register user* το οποίο σηματοδοτεί την εγγραφή ενός νέου χρήστη (user) με αναγνωριστικό <uid>. Το γεγονός αυτό προσθέτει το νέο χρήστη στη λίστα χρηστών της υπηρεσίας. Τα πεδία history και favorities του κόμβου που αντιστοιχεί στο χρήστη στη λίστα χρηστών πρέπει να έχουν την αρχική τιμή NULL.

Η εισαγωγή θα πρέπει να πραγματοποιείται σε χρόνο O(1). 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 στη λίστα χρηστών και για κάθε i ∈ {1, …, n}, <pid_i> είναι το αναγνωριστικό του χρήστη που αντιστοιχεί στον i-οστό κόμβο της λίστας αυτής.

– **U <uid>**

Γεγονός τύπου *unregister user* το οποίο σηματοδοτεί τη διαγραφή ενός χρήστη (user) με αναγνωριστικό <uid> από τη λίστα χρηστών. Πριν την οριστική διαγραφή του χρήστη από τη λίστα χρηστών θα πρέπει να διαγράψετε όλα τα στοιχεία της λίστα ιστορικού και της λίστας αγαπημένων ταινιών του χρήστη αν αυτές περιέχουν στοιχεία. 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 στη λίστα χρηστών και για κάθε i ∈ {1, …, n}, <pid_i> είναι το αναγνωριστικό του χρήστη που αντιστοιχεί στον i-οστό κόμβο της λίστας αυτής.

– **A <mid> <category> <year>**

Γεγονός τύπου *add new movie* ο οποίο σηματοδοτεί την άφιξη μιας νέας ταινίας που είναι διαθέσιμη στους χρήστες. Κατά το γεγονός αυτό θα δημιουργείτε μια νέα ταινία με αναγνωριστικό <mid> και έτος κυκλοφορίας <year>, η οποία θα ανήκει στη θεματική κατηγορία <category>. **Ανεξάρτητα από την κατηγορία στην οποία ανήκει, η νέα ταινία θα εισάγεται στη λίστα κατηγορίας «νέες κυκλοφορίες».** Η λίστα αυτή θα πρέπει να παραμένει ταξινομημένη μετά την κάθε εισαγωγή.

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

```
A
    MOVIES:
    <category_1>: <mid_1,1> ... <mid_1,n1>
    <category_2>: <mid_2,1> ... <mid_2,n2>
    ...
   <category_6>: <mid_6,1> ... <mid_6,n6>
DONE
```

όπου για κάθε i, $1 \leq i \leq 6$, n_i είναι το μέγεθος της λίστας ταινιών της i-οστής κατηγορίας του πίνακα κατηγοριών, και για κάθε j, $1 \leq j \leq n_i$, και $<mid_{i,j}>$ είναι το αναγνωριστικό της ταινίας του j-οστού κόμβου στη λίστα ταινιών της i-οστής κατηγορίας.

– **C**

Γεγονός τύπου *categorize movies* το οποίο σηματοδοτεί την ταξινόμηση των ταινιών που περιέχει η λίστα «νέες κυκλοφορίες» στις υπόλοιπες θεματικές κατηγορίες. Η διαδικασία αυτή θα πρέπει να εκτελείται σε χρόνο O(n), όπου n είναι ο αριθμός των ταινιών που περιέχονται στη λίστα «νέες κυκλοφορίες». Συγκεκριμένα, το γεγονός αυτό θα πρέπει να εκτελείτε με μια διάσχιση της λίστας ταινιών της κατηγορίας νέες κυκλοφορίες. Στο τέλος αυτής της διαδικασίας, όλες οι ταινίες από τη λίστα «νέες κυκλοφορίες» πρέπει να έχουν διαμοιραστεί στις υπόλοιπες κατηγορίες αδειάζοντας τη λίστα «νέες κυκλοφορίες».

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

```
C
    MOVIES:
    <category_1>: <mid_1,1> . . . <mid_1,n1>
    <category_2>: <mid_2,1> . . . <mid_2,n2>
    ...
   <category_6>: <mid_6,1> . . . <mid_6,n6>
DONE
```

όπου για κάθε i, $1 \leq i \leq 6$, n_i είναι το μέγεθος της λίστας ταινιών της i-οστής κατηγορίας του πίνακα κατηγοριών, και για κάθε j, $1 \leq j \leq n_i$, και $<mid_{i,j}>$ είναι το αναγνωριστικό της ταινίας του j-οστού κόμβου στη λίστα ταινιών της i-οστής κατηγορίας.

– **G <uid> <mid> <score>**

Γεγονός τύπου *rate movie* το οποίο σηματοδοτεί ότι ο χρήστης με αναγνωριστικό <uid> έχει παρακολουθήσει την ταινία με αναγνωριστικό <mid> και την αξιολογεί με βαθμό <score>. Κατά το γεγονός αυτό αρχικά θα γίνεται αναζήτηση της ταινίας με αναγνωριστικό <mid> σε όλες τις λίστες

ταινιών του πίνακα κατηγοριών. Στη συνέχεια θα δημιουργηθεί μια νέα εγγραφή τύπου «user_movie» χρησιμοποιώντας τις πληροφορίες τις ταινίας (movie) που έχει ήδη εντοπιστεί προηγουμένως και να αξιολογείται με βαθμό «score». Τέλος γίνεται η εισαγωγή της νέας «user_movie» στο ιστορικό του χρήστη με αναγνωριστικό <uid>. **Σημειώνεται ότι η λίστα ιστορικού του χρήστη πρέπει να παραμένει ταξινομημένη με βάση το πεδίο score.**

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

```
G <uid> <mid> <score>
      HISTORY: <mid_1, score_1>, ... , <mid_n, score_n>
DONE
```

όπου n είναι το μέγεθος της λίστας αγαπημένων ταινιών του χρήστη <uid> και για κάθε i ∈ {1, …, n}, $<mid_i>$ είναι το αναγνωριστικό της ταινίας και $<score_i>$ η βαθμολογία της ταινίας που αντιστοιχεί στο i-οστό κόμβο στη αγαπημένων του χρήστη <uid>.

– **F**

Γεγονός τύπου *identify favorites* το οποίο σηματοδοτεί την ενημέρωση της λίστας αγαπημένων ταινιών του κάθε εγγεγραμμένου χρήστη της υπηρεσίας. Κατά το γεγονός αυτό, όλες οι ταινίες του ιστορικού κάθε χρήστη, οι οποίες έχουν βαθμολογία μεγαλύτερη από ή ίση με 7 αντιγράφονται στη λίστα αγαπημένων του χρήστη.

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

```
F
      FAVORITES: <mid_1, score_1>, ... , <mid_n, score_n>
DONE
```

όπου n είναι το μέγεθος της λίστας αγαπημένων ταινιών του χρήστη <uid>, για κάθε i ∈ {1, …, n}, $<mid_i>$ είναι το αναγνωριστικό της ταινίας και $<score_i>$ η βαθμολογία της ταινίας που αντιστοιχεί στο i-οστό κόμβο στη αγαπημένων του χρήστη <uid>.

– **S <uid> <mid>**

Γεγονός τύπου *suggest movie* το οποίο αναζητά μια ταινία απ' τη λίστα των αγαπημένων κάποιου χρήστη (η οποία θα αποτελέσει την προτεινόμενη ταινία). Πιο συγκεκριμένα, κατά το γεγονός αυτό, γίνεται πρώτα αναζήτηση της ταινίας με αναγνωριστικό <mid> στη λίστα αγαπημένων του χρήστη με αναγνωριστικό <uid>. Στη συνέχεια, θα πρέπει να αναζητηθεί στη λίστα αγαπημένων ταινιών του ίδιου χρήστη μια ταινία (αν υπάρχει) που να ανήκει στην ίδια θεματική κατηγορία με την ταινία με αναγνωριστικό <mid> αλλά να έχει βαθμολογηθεί με μεγαλύτερο score από την ταινία με αναγνωριστικό <mid>.

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

```
S <uid> <mid>
    Primary movie: <mid_p> <score_p> <category_p>
    Suggested movie: <mid_S> <score_S> <category_S>
DONE
```

όπου <mid_p> <$score_p$> <$category_p$> είναι το ανγνωριστικό, η βαθμολογία και η κατηγορία, αντίστοιχα, της αρχικής ταινίας που αναζητήθηκε και <mid_S> <$score_S$> <$category_S$> είναι το ανγνωριστικό, η βαθμολογία και η κατηγορία, αντίστοιχα, της προτεινόμενης ταινίας.

– **I <mid>**

Γεγονός τύπου *search movie* το οποίο σηματοδοτεί την αναζήτηση της ταινίας με αναγνωριστικό <mid> σε όλες τις λίστες ταινιών του πίνακα κατηγοριών.

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

```
I <mid>
    <category>, <year>
DONE
```

όπου <category> είναι η κατηγορία που ανήκει η ταινία με αναγνωριστικό <mid> και <year> είναι το έτος κυκλοφορίας της.

– **M**

Γεγονός τύπου *print movies* το οποίο σηματοδοτεί την εκτύπωση της λίστας ταινιών όλων των κατηγοριών. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
M
MOVIES:
    <category_1>: <mid_1,1> . . . <mid_1,n1>
    <category_2>: <mid_2,1> . . . <mid_2,n2>
    ...
    <category_6>: <mid_6,1> . . . <mid_6,n6>
DONE
```

όπου για κάθε i, $1 \leq i \leq 6$, n_i είναι το μέγεθος της λίστας ταινιών της i-οστής κατηγορίας του πίνακα κατηγοριών, και για κάθε j, $1 \leq j \leq n_i$, και <$mid_{i,j}$> είναι το αναγνωριστικό της ταινίας του j-οστού κόμβου στη λίστα ταινιών της i-οστής κατηγορίας.

– P

Γεγονός τύπου *print users* το οποίο σηματοδοτεί την εκτύπωση της λίστας χρηστών. Για τον κάθε χρήστη θα πρέπει να εκτυπώνονται όλα τα στοιχεία του χρήστη, συμπεριλαμβανομένων της λίστας ιστορικού και της λίστας αγαπημένων ταινιών του χρήστη.

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

```
P

    USERS:

    <uid_1>
        HISTORY: <mid_1,1> … <mid_1,n1>

        FAVORITES: <mid_1,1> … <mid_1,m1>

    <uid_2>
        HISTORY: <mid_2,1> … <mid_2,n2>

        FAVORITES: <mid_2,1> … <mid_2,m2>

     ...

    <uid_k>
        HISTORY: <mid_κ,1> … <mid_κ,nk>

        FAVORITES: <mid_κ,1> … <mid_κ,mk>

DONE
```

όπου k είναι ο αριθμός των χρηστών στη λίστα χρηστών, για κάθε j, $1 \leq j \leq k$, n_j είναι το μέγεθος της λίστας ιστορικού του χρήστη με αναγνωριστικό $<uid_j>$ και m_j είναι το μέγεθος της λίστας αγαπημένων αυτού του χρήστη.

Βαθμολογία

R	Register user	7
U	Unregister user	7
A	Add new movie	10
C	Categorize movies	20
G	Rate movie	12
F	Identify favorites	15
S	Suggest movie	20
I	Search movie	3
M	Print movies	3
P	Print users	3
Δεν κάνει compile		-10
Δεν τρέχει και δεν τρέχουν τα test-files		-10

Δομές Δεδομένων

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

```
struct movie{
	int mid;
	int category;
	int year;
	struct movie *next;
};

struct user_movie {
	int mid;
	int category;
	int score;
	struct user_movie * next;
	struct user_movie * prev;
};

struct user {
	int uid;
	struct user_movie * history;
	struct user_movie * favorites;
	struct user * next;
};

/*The array of the movie categories*/
struct movie * Movie_categories[M];

/* global variable, pointer to the beginning of the users list*/
struct user * users_list;

/* global variable, pointer to the sentinel node of the users list */
struct user * users_sentinel;
```