HY240: Δομές Δεδομένων

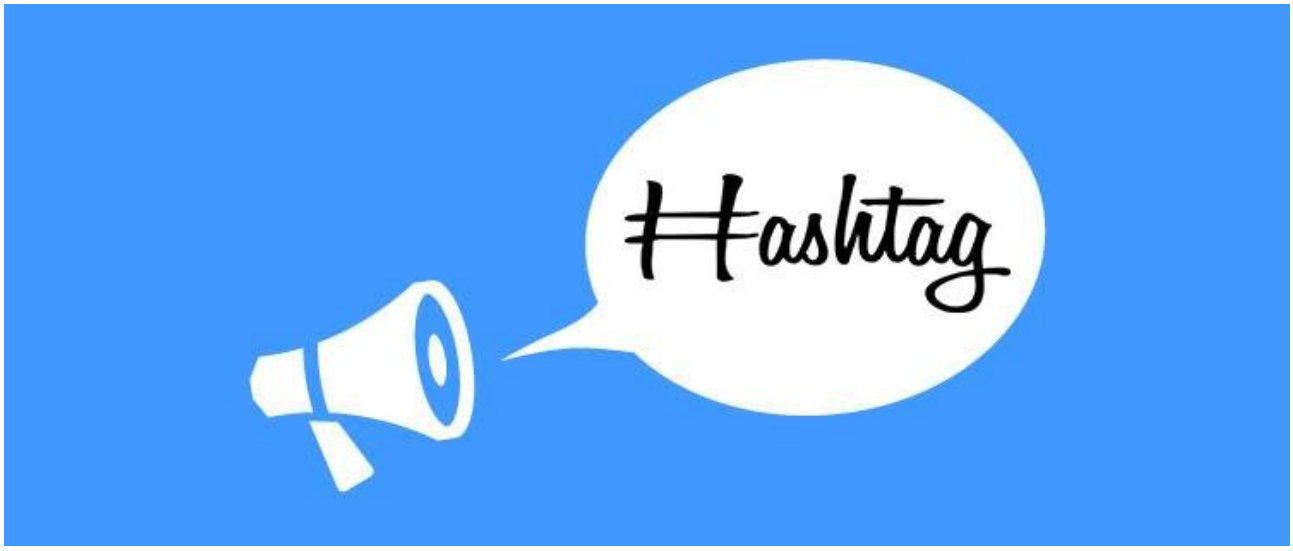
Εαρινό Εξάμηνο – Ακαδημαϊκό Έτος 2018

Διδάσκουσα: Παναγιώτα Φατούρου

Προγραμματιστική Εργασία - 1ο Μέρος

Ημερομηνία Παράδοσης: Κυριακή, 1 Απριλίου 2018, ώρα 23:59

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



Γενική Περιγραφή

Στη εργασία αυτή καλείστε να υλοποιήσετε ένα πρόγραμμα που να προσομοιώνει τη λειτουργία ενός μέσου κοινωνικής δικτύωσης παρόμοιο με το Twitter. Το πρόγραμμα που θα υλοποιήσετε θα πρέπει να επιτρέπει στους χρήστες του να στέλνουν και να διαβάζουν σύντομα μηνύματα, τα οποία ονομάζονται τουίτς (tweets). Ο κάθε χρήστης μπορεί να “ακολουθεί” (follow) άλλους χρήστες με σκοπό να εμφανίζονται τα tweets τους στην περιοχή του (Wall).

Αναλυτική Περιγραφή Ζητούμενης Υλοποίησης

Το σύστημα αποτελείται από ένα σύνολο χρηστών οι οποίοι αποθηκεύονται σε μία απλά συνδεδεμένη λίστα που ονομάζεται **λίστα χρηστών**. Τα στοιχεία της λίστας χρηστών είναι **ταξινομημένα, σε αύξουσα διάταξη,** με βάση το αναγνωριστικό του εκάστοτε χρήστη. Ο κάθε κόμβος της λίστας είναι μία εγγραφή τύπου struct user με τα ακόλουθα πεδία:

- uid: Αναγνωριστικό (τύπου int) που χαρακτηρίζει μοναδικά το χρήστη.
- followers: Δείκτης (τύπου struct follower) στο πρώτο στοιχείο μίας **απλά συνδεδεμένης λίστας** που ονομάζεται **λίστα οπαδών**.
- wall_head: Δείκτης (τύπου struct tweet_w) στο πρώτο στοιχείο μίας **απλά συνδεδεμένης λίστας** που ονομάζεται **λίστα των tweets** (Wall).
- wall_sentinel: Δείκτης (τύπου struct tweet_w) στον κόμβο φρουρό της λίστας των tweets του χρήστη.
- next: Δείκτης (τύπου struct user) στον επόμενο κόμβο της λίστας χρηστών.

Η λίστα οπαδών είναι **ταξινομημένη** με βάση το αναγνωριστικό του κάθε οπαδού. Κάθε στοιχείο της λίστας αυτής είναι μία εγγραφή τύπου struct follower με τα παρακάτω πεδία:

- uid: Αναγνωριστικό (τύπου int) που χαρακτηρίζει μοναδικά τον follower (user).
- next: Δείκτη (τύπου struct follower) στον επόμενο κόμβο της λίστας των followers.

Η λίστα των tweets είναι **μη ταξινομημένη, με κόμβο φρουρό.** Κάθε στοιχείο της λίστας αυτής είναι μία εγγραφή τύπου struct tweet_w με τα παρακάτω πεδία:

- tid: Αναγνωριστικό (τύπου int) που χαρακτηρίζει μοναδικά το tweet.
- uid: Αναγνωριστικό (τύπου int) που χαρακτηρίζει μοναδικά τον χρήστη που δημοσίευσε το tweet.
- next: Δείκτη (τύπου struct tweet_w) στον επόμενο κόμβο της λίστας των tweets.

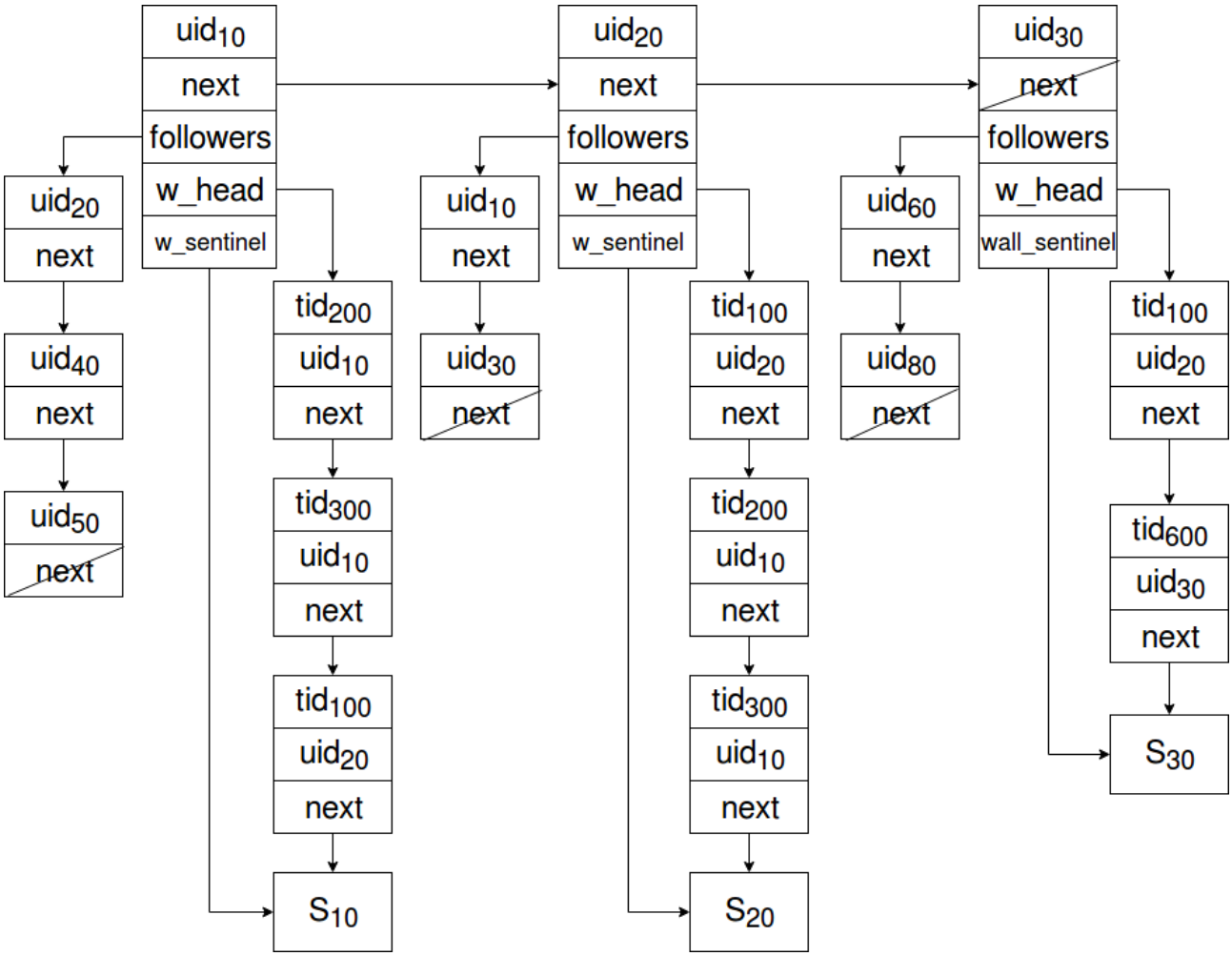
Στο Σχήμα 1 παρουσιάζεται η λίστα χρηστών, ο κάθε κόμβος της οποίας περιέχει μία λίστα followers και μία λίστα tweets.

Το σύστημα περιέχει επίσης και ένα μηχανισμό για κατηγοριοποίηση των tweets. Ο μηχανισμός αυτός υλοποιείται με ένα πίνακα Hashtags[MAX_TAGS] που ονομάζεται **πίνακας κατηγοριών**. Θεωρούμε ότι υπάρχουν 10 κατηγορίες tweets στο σύστημα και έτσι ο πίνακας έχει δέκα θέσεις, μια για κάθε κατηγορία. Επομένως, MAX_TAGS = 10. Το στοιχείο i του πίνακα είναι ένας δείκτης σε μία διπλά συνδεδεμένη λίστα που ονομάζεται **λίστα tweets της κατηγορίας i** και περιέχει όλα τα tweets με hashtag i. Η λίστα αυτή είναι **ταξινομημένη**, με βάση το πεδίο timestamp του καθε tweet. Ο κάθε κόμβος της λίστας είναι μία εγγραφή τύπου struct tweet_h με τα ακόλουθα πεδία:

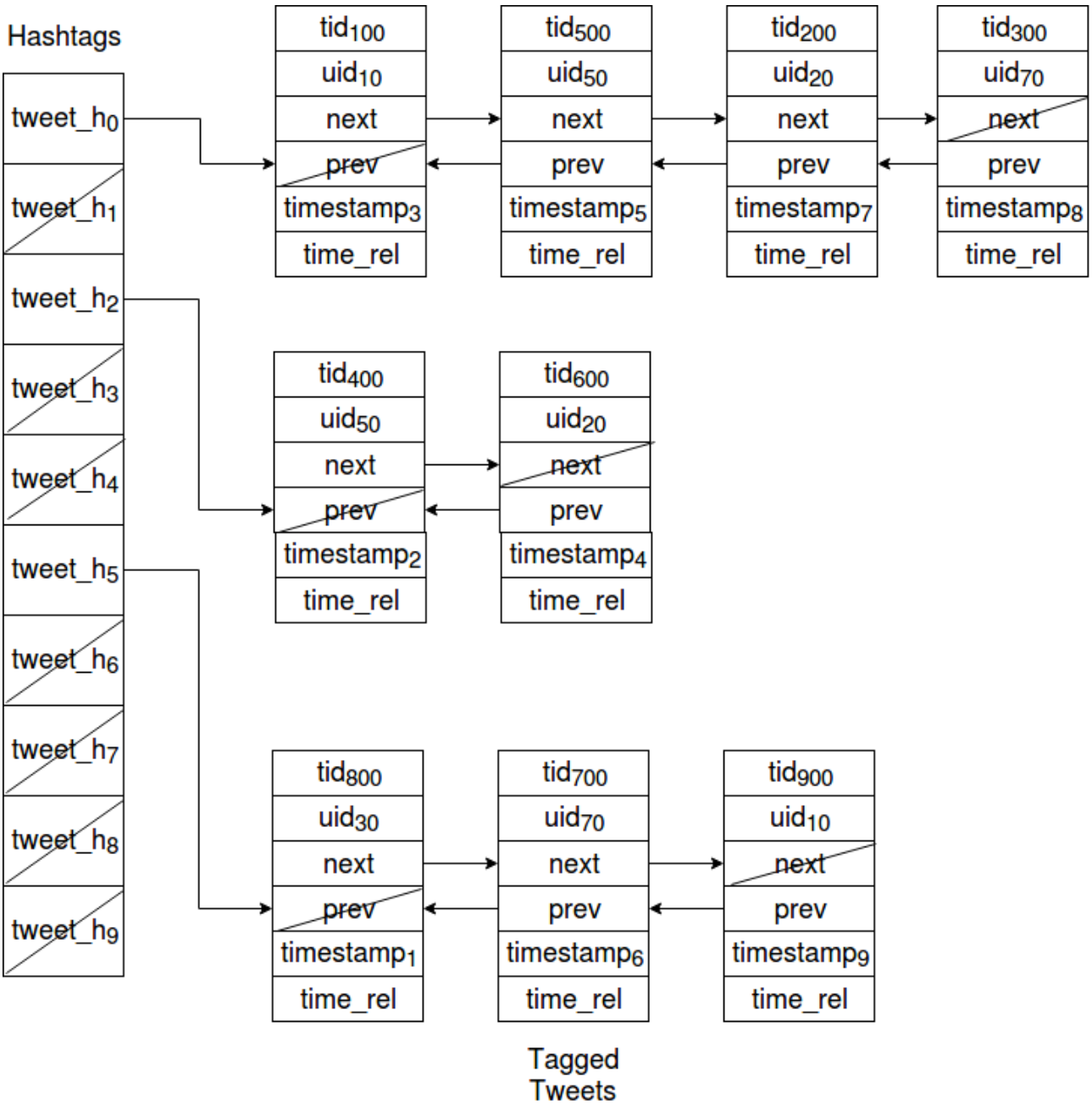
- tid: Αναγνωριστικό (τύπου int) που χαρακτηρίζει μοναδικά το tweet
- timestamp: Αναγνωριστικό (τύπου int) που αντιστοιχεί στην ημερομηνία δημοσίευσης του tweet. Το timestamp θα έχει τη μορφή YYYYMMDD. Για παράδειγμα, το timestamp 20180226 αντιστοιχεί στην ημερομηνία 26 Φεβρουαρίου 2018.

- time_relevance: Θετικός ακέραιος αριθμός που αντιστοιχεί στην χρονική ισχύ ενός tweet.
- next: Δείκτης (τύπου struct tweet_h) στον επόμενο κόμβο της λίστας των tagged tweets.
- prev: Δείκτης (τύπου struct tweet_h) στον προηγούμενο κόμβο της λίστας των tagged tweets.

Στο Σχήμα 2 παρουσιάζεται ο πίνακας κατηγοριών, το κάθε κελί του οποίου περιέχει μια λίστα από tagged tweets.



Σχήμα 1: Η απλά συνδεδεμένη λίστα χρηστών, η οποία είναι ταξινομημένη με βάση το αναγνωριστικό του κάθε χρήστη. Παρουσιάζεται επίσης η απλά συνδεδεμένη λίστα των followers για κάθε χρήστη, η οποία είναι ταξινομημένη με βάση το αναγνωριστικό του κάθε follower καθώς και η απλά συνδεδεμένη λίστα των tweets για κάθε χρήστη, η οποία είναι μη ταξινομημένη, με κόμβο φρουρό.



Σχήμα 2: Ο πίνακας κατηγοριών όπου κάθε στοιχείο του είναι ένα δείκτης σε μία διπλά συνδεδεμένη λίστα, η οποία περιέχει τα tweets που ανήκουν σ' αυτήν την κατηγορία και είναι ταξινομημένη ως προς το timestamp του κάθε tweet.

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* που υποδηλώνει την εισαγωγή ενός νέου χρήστη με αναγνωριστικό <uid> στο σύστημα. Κατά το γεγονός αυτό, θα γίνεται εισαγωγή ενός νέου κόμβου τύπου struct user στη λίστα χρηστών. Μετά από κάθε εισαγωγή, η λίστα χρηστών πρέπει να μένει ταξινομημένη. 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 θα κάθε $i \in \{1, ..., n\}$, $<uid_i>$ είναι το αναγνωριστικό του χρήστη που αντιστοιχεί στον i-οστό κόμβο της λίστας αυτής.

S $<uid_1>$ $<uid_2>$

Γεγονός τύπου *subscribe* που υποδηλώνει την εγγραφή του χρήστη με αναγνωριστικό $<uid_2>$ στη λίστα των followers του χρήστη με αναγνωριστικό $<uid_1>$. Κατά το γεγονός αυτό θα πρέπει να πραγματοποιούνται οι ακόλουθες ενέργειες. Αρχικά, θα αναζητήσετε στη λίστα χρηστών το χρήστη με αναγνωριστικό $<uid_1>$ και στη συνέχεια θα εισάγετε στη λίστα των followers ένα νέο κόμβο με αναγνωριστικό $<uid_2>$. Μετά από κάθε εισαγωγή, η λίστα των followers θα πρέπει να παραμένει ταξινομημένη. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

S $<uid_1>$ $<uid_2>$
    Followers = $<uid_1>$, $<uid_2>$, ..., $<uid_n>$
DONE

όπου n είναι ο αριθμός των κόμβων στη λίστα των followers του χρήστη με αναγνωριστικό $<uid_2>$ και για κάθε $i \in \{1, ..., n\}$, $<uid_i>$ είναι το αναγνωριστικό του χρήστη (follower) που αντιστοιχεί στον i-οστό κόμβο της λίστας αυτής.

T <uid> <tid> <hashtag> <timestamp> <time_relevance>

Γεγονός τύπου *tweet* που υποδηλώνει τη δημοσίευση ενός tweet με αναγνωριστικό <tid> από το χρήστη με αναγνωριστικό <uid>. Οι παράμετροι <timestamp> και <time_relevance> υποδηλώνουν τη χρονική στιγμή της δημοσίευσης και τη χρονική ισχύ του tweet, αντίστοιχα. Κατά το γεγονός αυτό, αρχικά θα πρέπει να αναζητήσετε στη λίστα χρηστών το χρήστη με αναγνωριστικό <uid> και στη συνέχεια θα εισάγετε στη λίστα των tweets του, ένα νέο κόμβο με αναγνωριστικό <tid>. Έπειτα, θα διατρέξετε τη λίστα των followers του χρήστη αυτού και για κάθε κόμβο στη λίστα, θα αναζητείτε το χρήστη με το τρέχον <uid> στη λίστα χρηστών. Αφού τον εντοπίσετε, θα εισάγετε στη λίστα των tweets του, ένα νέο κόμβο με αναγνωριστικό <tid>. Η διαδικασία δημοσίευσης του tweet σε όλες τις λίστες των followers του χρήστη που παράγει το tweet θα πρέπει να εκτελείται σε χρόνο O(n), όπου n είναι το πλήθος των κόμβων στη λίστα χρηστών. Θα πρέπει επομένως να διατρέξετε τη λίστα των χρηστών μόνο μία φορά. Τέλος, θα εισάγετε και ένα νέο κόμβο στον πίνακα κατηγοριών. Για την εισαγωγή αυτή, θα χρησιμοποιήσετε την παράμετρο <hashtag> του γεγονότος για να εντοπίσετε την κατάλληλη λίστα στον πίνακα κατηγοριών κι έπειτα θα εισάγετε ένα νέο κόμβο που να αντιπροσωπεύει το νέο tweet. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

T <uid> <tid> <hashtag> <timestamp> <time_relevance>
$User_1$ = $<tid_{1,1}:uid_{1,1}>$, $<tid_{1,2}:uid_{1,2}>$, ..., $<tid_{1,m1}:uid_{1,m1}>$
$User_2$ = $<tid_{2,1}:uid_{2,1}>$, $<tid_{2,2}:uid_{2,2}>$, ..., $<tid_{2,m2}:uid_{2,m2}>$
...
$User_n$ = $<tid_{n,1}:uid_{n,1}>$, $<tid_{n,2}:uid_{n,2}>$, ..., $<tid_{n,mn}:uid_{n,mn}>$
DONE

για κάθε i, $1 \le i \le n$, m_i είναι το πλήθος των κόμβων της λίστας των tweets του i-οστού χρήστη, και για κάθε $j \in \{1, ..., m_i\}$, $tid_{i,j}$, $uid_{i,j}$ είναι το αναγνωριστικό του tweet και το αναγνωριστικό του χρήστη, αντίστοιχα, που αντιστοιχεί στον j-οστό κόμβο της λίστας των tweets του i-οστού χρήστη.

U <uid_1> <uid_2>

Γεγονός τύπου *unsubscribe* που υποδηλώνει τη διαγραφή του χρήστη με αναγνωριστικό <uid_2> από τη λίστα των followers του χρήστη με αναγνωριστικό <uid_1>. Κατά το γεγονός αυτό, θα πρέπει να αναζητήσετε στη λίστα χρηστών το χρήστη με αναγνωριστικό <uid_1> και στη συνέχεια να αφαιρέσετε από τη λίστα των followers του χρήστη με αναγνωριστικό <uid_1>, το χρήστη με αναγνωριστικό <uid_2>. Έπειτα, θα πρέπει να διατρέξετε τη λίστα των tweets του χρήστη με αναγνωριστικό <uid_2> και να αφαιρέσετε από αυτή τους κόμβους που έχουν το πεδίο uid ίσο με <uid_1> (δηλαδή θα πρέπει να σβήσετε από τη λίστα των tweets του χρήστη με αναγνωριστικό <uid_2> τα tweets που έχουν παραχθεί από το χρήστη με αναγνωριστικό <uid_1>). Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

U <uid_1> <uid_2>
$User_1$ = <$tid_{1,1}$:$uid_{1,1}$>, <$tid_{1,2}$:$uid_{1,2}$>, ..., <$tid_{1,m1}$:$uid_{1,m1}$>
$User_2$ = <$tid_{2,1}$:$uid_{2,1}$>, <$tid_{2,1}$:$uid_{2,2}$>, ..., <$tid_{2,m2}$:$uid_{2,m2}$>
...
$User_n$ = <$tid_{n,1}$:$uid_{n,1}$>, <$tid_{n,2}$:$uid_{n,2}$>, ..., <$tid_{n,mn}$:$uid_{n,mn}$>
DONE

για κάθε i, $1 \le i \le n$, m_i είναι το πλήθος των κόμβων της λίστας των tweets του i-οστού χρήστη, και για κάθε $j \in \{1, ..., m_i\}$, $tid_{i,j}$ και $uid_{i,j}$, είναι το αναγνωριστικό του tweet και το αναγνωριστικό του χρήστη του tweet αντίστοιχα, που αντιστοιχεί στον j-οστό κόμβο της λίστας των tweets του i-οστού χρήστη.

D <uid>

Γεγονός τύπου *delete user* που υποδηλώνει τη διαγραφή του χρήστη με αναγνωριστικό <uid> από το σύστημα. Κατά το γεγονός αυτό, θα πρέπει να αναζητήσετε στη λίστα χρηστών το χρήστη με αναγνωριστικό <uid> και να διαγράψετε όλα τα στοιχεία της λίστας tweets αυτού του χρήστη. Στη συνέχεια, θα πρέπει να διατρέξετε τη λίστα των οπαδών και για κάθε κόμβο θα πρέπει να καλείτε τη συνάρτηση που υλοποιεί το γεγονός *unsubscribe*, δίνοντας ως ορίσματα: <uid>, <uid_i>, όπου <uid_i> είναι ο τρέχων κόμβος που διασχίζετε στη λίστα οπαδών. Τέλος, θα πρέπει να διαγράφετε από τη λίστα χρηστών το χρήστη με αναγνωριστικό <uid>. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

D <uid>
$User_1$ = <$tid_{1,1}$:$uid_{1,1}$>, <$tid_{1,2}$:$uid_{1,2}$>, ..., <$tid_{1,m1}$:$uid_{1,m2}$>
$User_2$ = <$tid_{2,1}$:$uid_{2,1}$>, <$tid_{2,1}$:$uid_{2,2}$>, ..., <$tid_{2,m2}$:$uid_{2,m2}$>
...
$User_n$ = <$tid_{n,1}$:$uid_{n,1}$>, <$tid_{n,2}$:$uid_{n,2}$>, ..., <$tid_{n,mn}$:$uid_{n,mn}$>
DONE

για κάθε i, $1 \le i \le n$, m_i είναι το πλήθος των κόμβων της λίστας των tweets του i-οστού χρήστη, και για κάθε $j \in \{1, ..., m_i\}$, $tid_{i,j}$ και $uid_{i,j}$ είναι το αναγνωριστικό του tweet, το αναγνωριστικό του χρήστη του tweet αντίστοιχα, που αντιστοιχεί στον j-οστό κόμβο της λίστας των tweets του i-οστού χρήστη.

L <uid> <tid>

Γεγονός τύπου *lookup* που υποδηλώνει την αναζήτηση ενός tweet με αναγνωριστικό <tid> στη λίστα των tweets του χρήστη με αναγνωριστικό <uid>. Κατά το γεγονός αυτό, θα αναζητήσετε στη λίστα χρηστών το χρήστη με αναγνωριστικό <uid> και στη συνέχεια θα αναζητήσετε στη λίστα των tweets του τον κόμβο με αναγνωριστικό <tid>. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
L <uid> <tid>
        Tweet = <tid:uid>
DONE
```

όπου tid και uid είναι το αναγνωριστικό του tweet και το αναγνωριστικό του χρήστη στον οποίο ανήκει το tweet.

M <hashtag$_1$> <hashtag$_2$>

Γεγονός τύπου *merge* που υποδηλώνει τη συγχώνευση των tweets στον πίνακα Hashtags που ανήκουν στην κατηγορία <hashtag$_1$> ή <hashtag$_2$>. Κατά το γεγονός αυτό, θα αναζητήσετε στον πίνακα Hashtags τις δύο λίστες tweets που αντιστοιχούν στα παραπάνω hashtags και στη συνέχεια θα συνενώσετε τις λίστες αυτές με βάση το πεδίο timestamp. Το αποτέλεσμα της συνένωσης μπορεί απλά να τυπώνεται (δεν χρειάζεται να το αποθηκεύεσετε σε κάποια άλλη δομή). Η διαδικασία αυτή θα πρέπει να ολοκληρώνεται σε χρονική πολυπλοκότητα O(n1+n2), όπου n1 και n2 είναι το πλήθος των κόμβων των λιστών tweets με hashtags <hashtag$_1$> και <hashtag$_2$>, αντίστοιχα. Θα πρέπει δηλαδή, ο αλγόριθμός σας να διατρέχει καθεμιά από τις δύο αυτές λίστες μονάχα μία φορά προκειμένου να τυπώσει το αποτέλεσμα της συνένωσης. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
M <hashtag1> <hashtag2>
        Merged = <tid1:uid1:ts1:tm1>,...,<tidn1+n2:uidn1+n2:tsn1+n2:tmn1+n2>
        Tweets1 = <tid1,1:uid1,1:ts1,1:tm1,1>,...,<tid1,n1:uid1,n1:ts1,n1:tm1,n1>
        Tweets2 = <tid2,1:uid2,1:ts2,1:tm2,1>,...,<tid2,n2:uid2,n2:ts2,n2:tm2,n2>
DONE
```

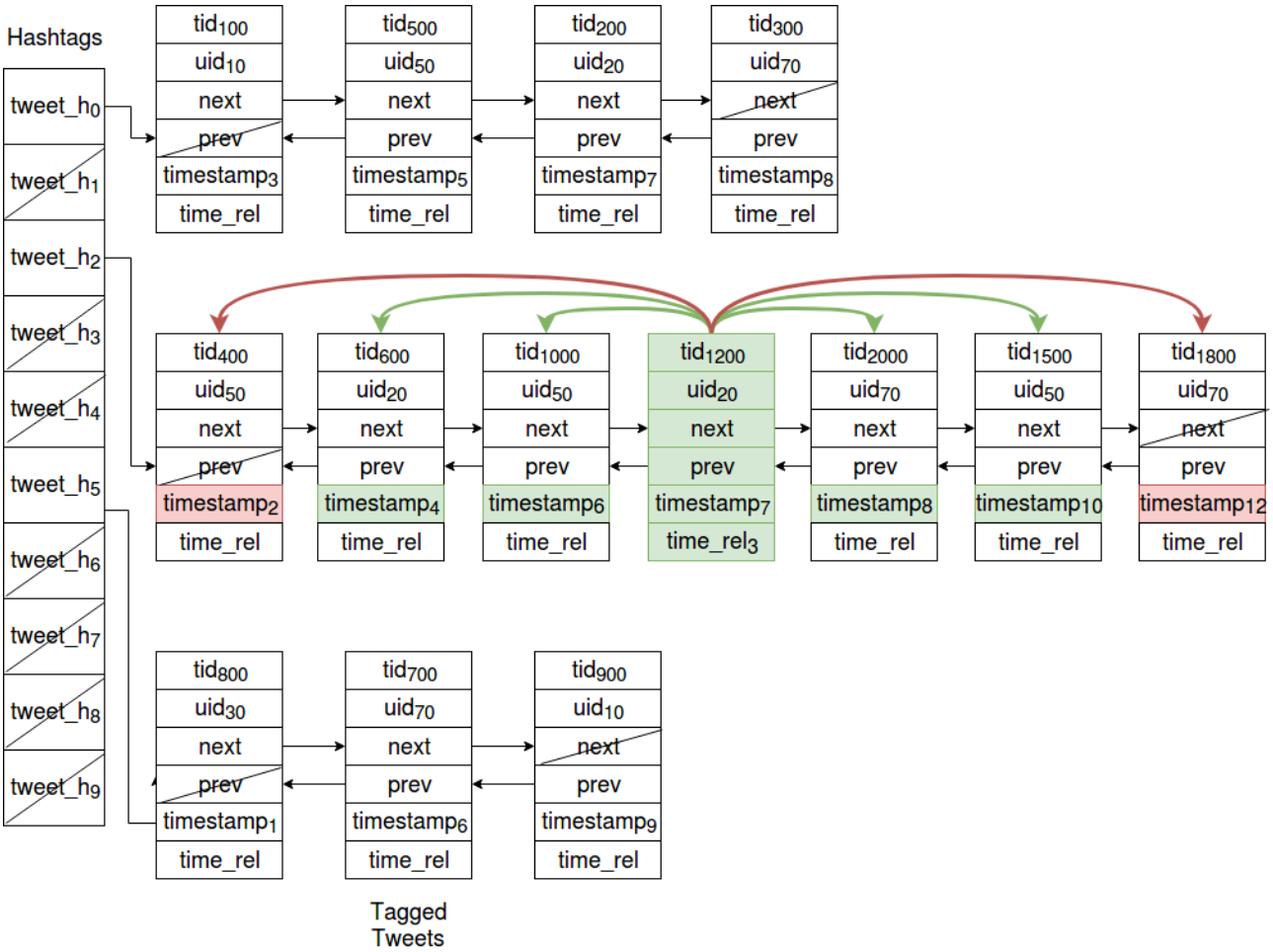
για κάθε i, $1 \leq i \leq 2$, n_i είναι το πλήθος των κόμβων της λίστας tweets του <hashtag$_i$> και για κάθε j $\in \{1, ..., n_i\}$, $tid_{i,j}$, $uid_{i,j}$, $ts_{i,j}$ και $tm_{i,j}$ είναι το αναγνωριστικό του tweet, το αναγνωριστικό του χρήστη, το timestamp και το time_relevance του tweet που αντιστοιχεί στον j-οστό κόμβο της λίστας των tagged tweets του i-οστού hashtag και για κάθε k, $1 \leq k \leq n_1+n_2$, tid_k, uid_k, ts_k και tm_k είναι το αναγνωριστικό του tweet, το αναγνωριστικό του χρήστη, το timestamp και το time_relevance του tweet που αντιστοιχεί στον κ-οστό κόμβο της συνένωσης των δύο λιστών των tagged tweets.

F <tid> <hashtag>

Γεγονός τύπου *time relevant tweets* που υποδηλώνει την αναζήτηση των tweets που ανήκουν στην κατηγορία <hashtag>, τα οποία ήταν στην επικαιρότητα (trend) την ίδια χρονική περίοδο με το tweet με αναγνωριστικό <tid>. Κατά το γεγονός αυτό, θα αναζητήσετε στον πίνακα Hashtags τη λίστα των tweets της κατηγορίας <hashtag> και στη συνέχεια θα εντοπίσετε το tweet με αναγνωριστικό <tid> διασχίζοντας τη λίστα αυτή. Έπειτα, θα εξετάσετε το πεδίο <time_relevance> του tweet και ξεκινώντας από το tweet αυτό, θα κάνετε κατάλληλη διάσχισηzig-zag στη λίστα προκειμένου να τυπώσετε τα tweets των οποίων το πεδίο <timestamp> έχει τιμή στο διάστημα

[timestamp – time_relevance, timestamp + time_relevance], όπου timestamp και time_relevance είναι οι τιμές των αντίστοιχων πεδίων του tweet με αναγνωριστικό <tid>.

Στο Σχήμα 3, παρουσιάζεται η εκτέλεση του γεγονότος R 1200 2. Το tweet με αναγνωριστικό 1200 και hashtag 2, έχει χρονική ισχύ 3. Ξεκινώντας λοιπόν από αυτό τον κόμβο της λίστας, διασχίζουμε τη λίστα προς τα αριστερά και παρατηρούμε πως τα tweets με αναγνωριστικό 1000 και 600 έχουν τιμή στο πεδίο timestamp 6 και 4 αντίστοιχα, τιμές που ανήκουν στο διάστημα [7-3, 7+3] (ή [4, 10]). Ο κόμβος με αναγνωριστικό 400 έχει timestamp 2, το οποίο δεν ανήκει στο διάστημα [4, 10]. Ανάλογα, θα διασχίσουμε τη λίστα προς τα δεξιά, και θα δούμε πως οι κόμβοι με αναγνωριστικό 2000 και 1500 έχουν τιμές στο πεδίο timestamp $8 \in [4,10]$ και $10 \in [4,10]$ αντίστοιχα, ενώ ο κόμβος με αναγνωριστικό 1800 έχει timestamp $12 \notin [4,10]$.



Σχήμα 3: Ο πίνακας κατηγοριών κατά την εκτέλεση του γεγονότος R 1200 2. Με πράσινο αποτυπώνονται οι τα tweets που είναι *time relevant* με το tweet με αναγνωριστικό 1200, ενώ με κόκκινο τα tweets που είναι είναι time irrelevant.

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

```
F <tid> <hashtag>
     Relevant Tweets = <tid1:uid1:ts1:tm1>, ..., <tidn:uidn:tsn:tmn>
DONE
```

όπου n είναι ο αριθμός των κόμβων στη λίστα των tweets της κατηγορίας <hashtag>, τα οποία είναι time relevant με το tweet με αναγνωριστικό <tid> και για κάθε i ϵ {1, ..., n}, tid_i, uid_i, ts_i και tm_i είναι το αναγνωριστικό του tweet, το αναγνωριστικό του χρήστη, το timestamp και το time_relevance του tweet που αντιστοιχεί στον i-οστό κόμβο της λίστας των tweets της κατηγορίας <hashtag> που είναι time relevant με το tweet με αναγνωριστικό <tid>.

X

Γεγονός τύπου *print users* το οποίο σηματοδοτεί την εκτύπωση όλων των χρηστών από τη λίστα χρηστών. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
X
    User1 = uid1
      Followers1: <uid1>, <uid2>, ..., <uidn1>
      Tweets1: <tid1>, <tid2>, ..., <tidm1>
    ...
    Userk = uidk
      Followersk: <uid1>, <uid2>, ..., <uidnk>
      Tweetsk: <tid1>, <tid2>, ..., <tidmk>
DONE
```

όπου k είναι το πλήθος των κόμβων στην λίστα των χρηστών, για κάθε i, $1 \leq i \leq k$, n_i είναι το πλήθος των κόμβων της λίστας των followers του i-οστού χρήστη και για κάθε $j \in \{1, ..., n_i\}$, uid_j είναι το αναγνωριστικό του j-οστού κόμβου της λίστας των followers του i-οστού χρήστη. Τέλος, m_i είναι το πλήθος των κόμβων της λίστας των tweets του i-οστού χρήστη και για κάθε $z \in \{1, ..., m_i\}$, tid_z είναι το αναγνωριστικό του z-οστού κόμβου της λίστας των tweets του i-οστού χρήστη.

Y

Γεγονός τύπου *print tweets* το οποίο σηματοδοτεί την εκτύπωση του πίνακα Hashtags. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
Y
  Hashtag1 = <tid1,1:uid1,1:ts1,1:tm1,1>,...,<tid1,n1:uid1,n1:ts1,n1:tm1,n1>
  Hashtag2 = <tid2,1:uid2,1:ts2,1:tm2,1>,...,<tid2,n2:uid2,n2:ts2,n2:tm2,n2>
  ...
  Hashtagk = <tidn,1:uidn,1:tsn,1:tmn,1>,...,<tidn,nk:uidn,nk:tsn,nk:tmn,nk>
DONE
```

για κάθε j, $1 \leq j \leq k$, n_j είναι το πλήθος των κόμβων της λίστας tweets της j-οστής κατηγορίας και για κάθε i ϵ {1, ..., n_j}, $tid_{j,i}$, $uid_{j,i}$, $ts_{j,i}$ και $tm_{j,i}$ είναι το αναγνωριστικό του tweet, το αναγνωριστικό του χρήστη, το timestamp και το time_relevance του tweet που αντιστοιχεί στον i-οστό κόμβο της λίστας tweets της j-οστής κατηγορίας.

Βαθμολογία Γεγονότων

R	10
S	10
T	15
U	15
D	15
L	5
M	12
F	12
X	3
Y	3

Δομές Δεδομένων

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

```
struct user {
        int uid;
        struct follower *followers;
        struct tweet_w *wall_head;
        struct tweet_w *wall_sentinel;
        struct user *next;
};

struct tweet_w {
        unsigned int  tid;
        unsigned int  uid;
        struct tweet_w *next;
};

struct follower {
        int uid;
        struct follower *next;
};

struct tweet_h {
        int tid;
        int uid;
        int timestamp;
        int time_relevance;
        struct tweet_h *next;
        struct tweet_h *prev;
};
```

```
/* global variable, pointer to the head of the users list */
struct user *usersList;

/* global enum, used to specify tweets hashtags */
typedef enum {
        sports,
        politics,
        economics,
        music,
        movies,
        nature,
        art,
        environment,
        technology,
        weather
}hashtag;

/* global variable, array of lists of tagged tweets */
struct tweet_h *Hashtags[10];
```