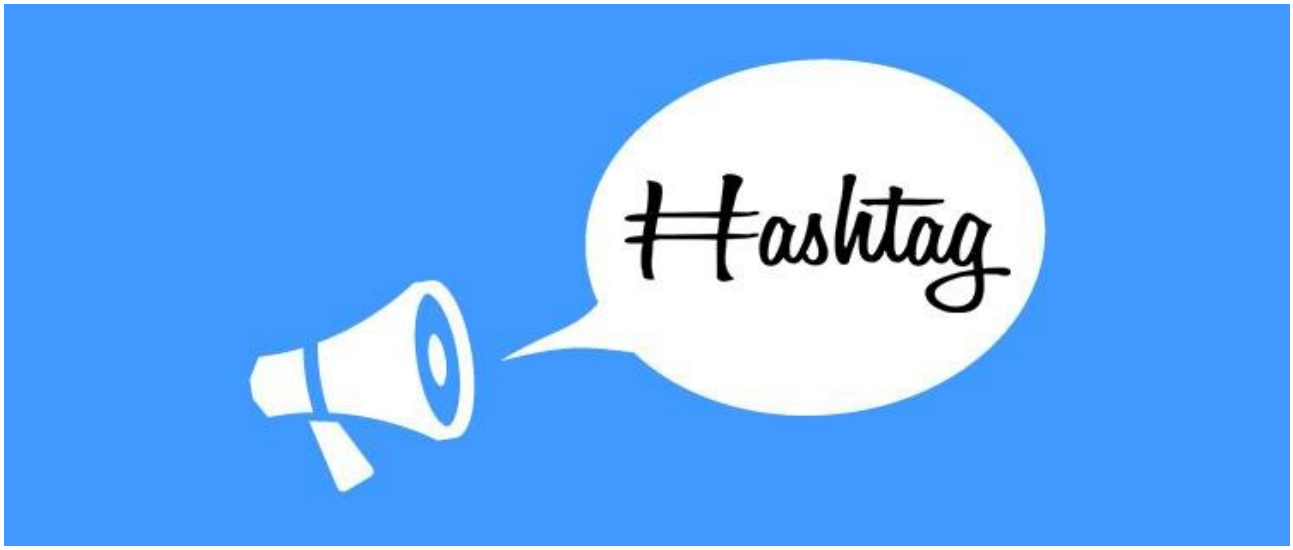
ΗΥ240: Δομές Δεδομένων
Εαρινό Εξάμηνο – Ακαδημαϊκό Έτος 2018
Διδάσκουσα: Παναγιώτα Φατούρου
Προγραμματιστική Εργασία - 2ο Μέρος

Ημερομηνία Παράδοσης: Δευτέρα, 14 Μαΐου 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

Στην 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 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 ένα **διπλά συνδεδεμένο δυαδικό δένδρο αναζήτησης (binary search tree), με κόμβο φρουρό** που ονομάζεται **δένδρο χρηστών**. Το δένδρο χρηστών είναι ταξινομημένο με βάση το αναγνωριστικό του χρήστη. Κάθε κόμβος του περιέχει έναν δείκτη προς τον γονικό του κόμβο. Όλα οι δείκτες που δεν δείχνουν σε θυγατρικούς ή γονικούς κόμβους στο δένδρο (δηλαδή όλοι οι δείκτες που θα ήταν NULL) δείχνουν στον κόμβο φρουρό. Ο κάθε κόμβος του δένδρου είναι μια εγγραφή τύπου **struct Tree_user** με τα ακόλουθα πεδία :

- **uid :** Αναγνωριστικό (τύπου int) που χαρακτηρίζει μοναδικά το χρήστη.

- **followers_R:** Δείκτης (τύπου struct TREE_followers *) που δείχνει στον κόμβο ρίζα ενός **απλού δυαδικού δένδρου αναζήτησης,** που ονομάζεται **δένδρο οπαδών**.

- **wall_R:** Δείκτης (τύπου struct TREE_tweet_w *) που δείχνει στον κόμβο ρίζα ενός **εμπλουτισμένου φυλλοπροσανατολισμένου δένδρου δυαδικής αναζήτησης,** που ονομάζεται **δένδρο των tweets** (Wall).

- **p_node:** Δείκτης (τύπου struct TREE_user *) που δεικτοδοτεί τον γονικό κόμβο του κόμβου με αναγνωριστικό uid.

- **lc:** Δείκτης (τύπου struct TREE_user *) που δεικτοδοτεί τον **αριστερό θυγατρικό κόμβο** του κόμβου με αναγνωριστικό uid.

- **rc:** Δείκτης (τύπου struct TREE_user *) που δεικτοδοτεί το **δεξιό θυγατρικό κόμβο** του κόμβου με αναγνωριστικό uid.

Το δένδρο οπαδών είναι **ταξινομημένο** ως προς το αναγνωριστικό του κάθε οπαδού. Κάθε στοιχείο του δένδρου αυτού είναι μια εγγραφή τύπου **struct TREE_followers** με τα παρακάτω πεδία:

- **uid :** Αναγνωριστικό (τύπου int) που χαρακτηρίζει μοναδικά τον οπαδό (user).

- **lc:** Δείκτης (τύπου struct TREE_followers *) που δεικτοδοτεί τον **αριστερό θυγατρικό κόμβο** του κόμβου με αναγνωριτσικό uid.

- **rc:** Δείκτης (τύπου struct TREE_followers *) που δεικτοδοτεί τον **δεξιό θυγατρικό κόμβο** του κόμβου με αναγνωριτσικό uid.

Το δένδρο των tweets είναι ένα **εμπλουτισμένο φυλλοπροσανατολισμένο δένδρο δυαδικής αναζήτησης, ταξινομημένο** ως προς την ημερομηνία δημοσίευσης του tweet (**timestamp**). Ένα εμπλουτισμένο φυλλοπροσανοτολισμένο δένδρο είναι ένα φυλλοπροσανατολισμένο δένδρο του οποίου όλα τα φύλλα είναι συνδεδεμένα ώστε να

σχηματίζουν μια ταξινομημένη απλά συνδεδεμένη λίστα. Επομένως, υπάρχει ένας επιπρόσθετος δείκτης next σε κάθε κόμβο. Ο δείκτης αυτός είναι NULL αν ο κόμβος είναι εσωτερικός, ενώ δείχνει στο φύλλο με το αμέσως μεγαλύτερο κλειδί διαφορετικά. Κάθε στοιχείο του δένδρου αυτού είναι μια εγγραφή τύπου struct TREE_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.
- **timestamp:** Ακέραιος (τύπου int) που αντιστοιχεί στην ημερομηνία δημοσίευσης του tweet. Κατ' αντιστοιχία με το πρώτο μέρος της προγραμματιστικής εργασίας, το πεδίο timestamp θα έχει τη μορφή YYYYMMDD. Για παράδειγμα, το timestamp 20180402 αντιστοιχεί στην ημερομηνία 02 Απριλίου 2018.
- **lc:** Δείκτης (τύπου struct TREE_tweet_w *) που δεικτοδοτεί τον **αριστερό θυγατρικό κόμβο** του κόμβου με αναγνωριστικό **tid**.
- **rc:** Δείκτης (τύπου struct TREE_tweet_w *) που δεικτοδοτεί το **δεξιό θυγατρικό κόμβο** του κόμβου με αναγνωριστικό **tid**.
- **next:** Δείκτης (τύπου struct TREE_tweet_w *) που δεικτοδοτεί το **επόμενο φύλλο** (δηλαδή το φύλλο με το αμέσως **μεγαλύτερο κλειδί** από το κλειδί του τρέχοντος φύλλου).

Στο **Σχήμα 1** παρουσιάζεται το δένδρο χρηστών και στο **Σχήμα 2** παρουσιάζεται ένας κόμβος του δένδρου χρηστών. Κατάλληλα πεδία αυτού του κόμβου είναι δείκτες προς τους ριζικούς κόμβους των δένδρων followers και tweets του χρήστη.

Το σύστημα περιέχει επίσης και ένα μηχανισμό όπου κρατά πληροφορίες για όλα τα tweets του συστήματος. Ο μηχανισμός αυτός υλοποιείται με ένα **πίνακα κατακερματισμού** που ονομάζεται **πίνακας κατακερματισμού tweets.** Η **επίλυση των συγκρούσεων γίνεται βάσει της τεχνικής του διπλού κατακερματισμού με ανοικτή διευθυνσιοδότηση (double hashing)**. Το μέγεθος του πίνακα κατακερματισμού θα πρέπει να επιλέγεται από εσάς προσεκτικά και θα πρέπει να είστε σε θέση να δικαιολογήσετε την επιλογή σας. Το μέγιστο πλήθος tweets που μπορεί να υπάρχουν στο σύστημα είναι 1000. Μέσω της μεταβλητής (max_tweets_g) θα καθορίσετε το μέγεθος του πίνακα κατακερματισμού που επιλέξατε. Το μέγεθος του hashTable αυτού θα πρέπει να επιλέγεται από ένα πίνακα πρώτων αριθμών (int primes_g[160]). Το κάθε tweet αποτελεί μια εγγραφή τύπου tweet (struct tweet) με τα ακόλουθα πεδία :

- **tid :** Αναγνωριστικό (τύπου int) που χαρακτηρίζει μοναδικά το tweet.

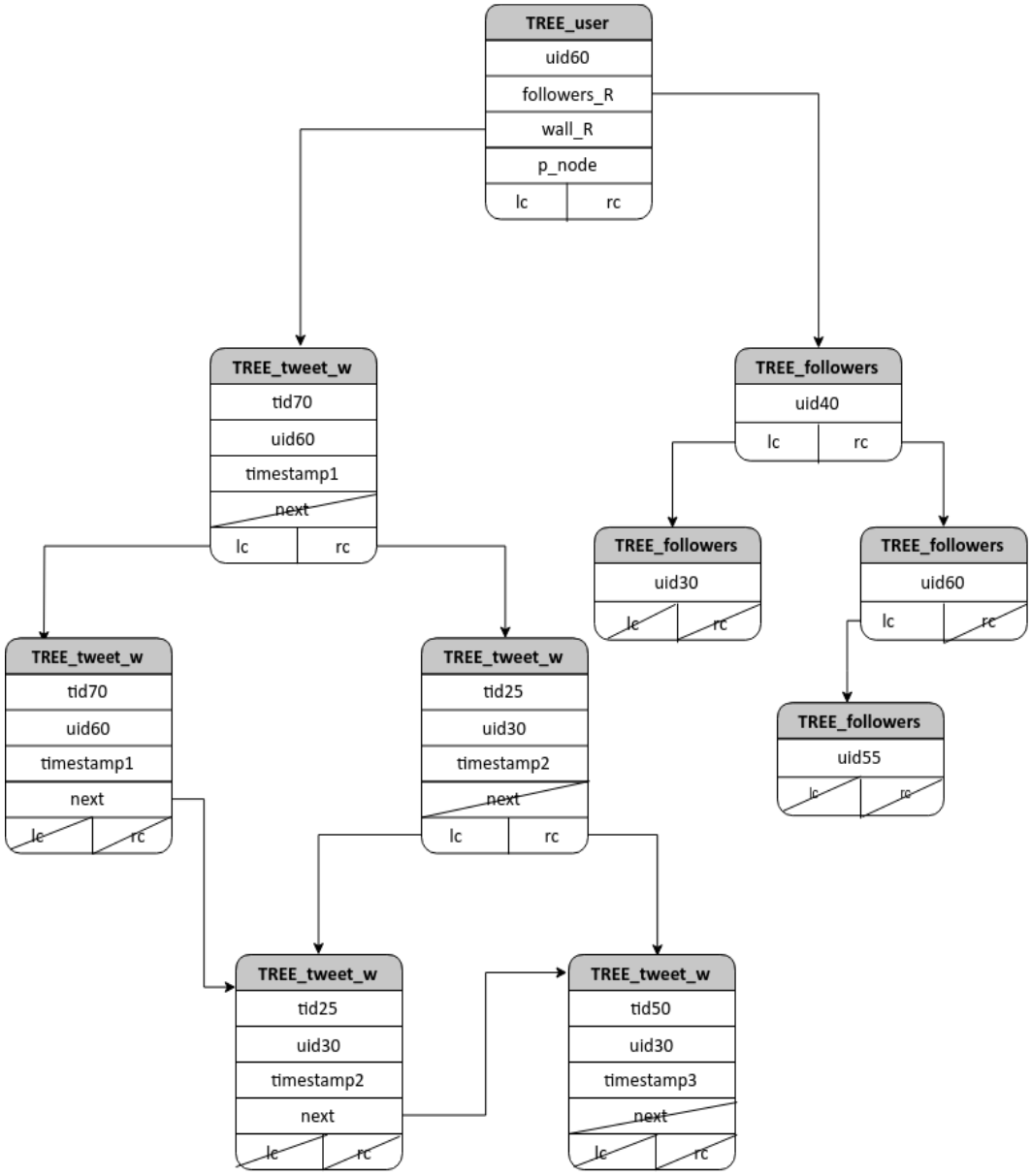
- **uid:** Αναγνωριστικό (τύπου int) που χαρακτηρίζει μοναδικά τον χρήστη που δημοσίευσε το tweet.

- **timestamp:** Ακέραιος (τύπου int) που αντιστοιχεί στην ημερομηνία δημοσίευσης του tweet. Το timestamp θα έχει τη μορφή YYYYMMDD. Για παράδειγμα, το timestamp 20180402 αντιστοιχεί στην ημερομηνία 02 Απριλίου 2018.

- **delete**: Ένα bit (που μπορεί να υλοποιηθεί ως char) το οποίο υποδηλώνει αν το tweet έχει διαγραφεί ή όχι.

Στο **Σχήμα 3** παρουσιάζεται ένα στιγμιότυπο του πίνακα κατακερματισμού με κάποιες εγγραφές μέσα.

Σχήμα 1 : Δένδρο Χρηστών. Κάθε κόμβος του δένδρου περιέχει πεδία που αποθηκεύουν δείκτες στους ριζικούς κόμβους των δένδρων οπαδών και tweets του χρήστη στον οποίο αντιστοιχεί ο κόμβος.



Σχήμα 2 : Παρουσιάζεται ως παράδειγμα ένας κόμβος του δένδρου χρηστών όπου δείχνει σε ένα δένδρο οπαδών και σε ένα δένδρο tweets.

Σχήμα 3: Παρουσιάζεται ένα στυγμιότυπο του πίνακα κατακερματισμού των tweets .

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 TREE_user στο δένδρο χρηστών. Μετά από κάθε εισαγωγή, το δένδρο χρηστών πρέπει να παραμένει ταξινομημένο. 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 στο δένδρο χρηστών και για κάθε i ∈ {1, ..., n}, <uid'$_i$> είναι το αναγνωριστικό του χρήστη που αντιστοιχεί στον i-οστό κόμβο του δένδρου χρηστών (βάσει της ενδοδιατεταγμένης διάσχισης).

S <uid$_1$> <uid$_2$>

Γεγονός τύπου *subscribe* που υποδηλώνει την εγγραφή του χρήστη με αναγνωριστικό <uid$_2$> στο δένδρο οπαδών του χρήστη με αναγνωριστικό <uid$_1$>. Κατά το γεγονός αυτό θα πρέπει να πραγματοποιούνται οι ακόλουθες ενέργειες. Αρχικά, θα αναζητήσετε στο δένδρο χρηστών το χρήστη με αναγνωριστικό <uid$_1$> και στη συνέχεια θα εισάγετε στο δένδρο οπαδών αυτού του χρήστη ένα νέο κόμβο με αναγνωριστικό <uid$_2$>. Μετά από κάθε εισαγωγή, το δένδρο οπαδών θα πρέπει να παραμένει ταξινομημένο. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
S <uid1> <uid2>
        Followers = <uid'1>, <uid'2>, ..., <uid'n>
DONE
```

όπου n είναι ο αριθμός των κόμβων στο δένδρο των followers του χρήστη με αναγνωριστικό <uid$_1$> και για κάθε i ∈ {1, ..., n}, <uid'$_i$> είναι το αναγνωριστικό του χρήστη (follower) που αντιστοιχεί στον i-οστό κόμβο του δένδρου (βάσει της ενδοδιατεταγμένης διάσχισης).

T <uid> <tid> <timestamp>

Γεγονός τύπου *tweet* που υποδηλώνει τη δημοσίευση ενός tweet με αναγνωριστικό <tid> από το χρήστη με αναγνωριστικό <uid>. Η παράμετρος <timestamp> υποδηλώνει τη χρονική στιγμή της δημοσίευσης. Κατά το γεγονός αυτό, αρχικά θα πρέπει να αναζητήσετε το χρήστη με αναγνωριστικό <uid> στο δένδρο χρηστών και στη συνέχεια θα πρέπει να εισάγετε στο δένδρο των tweets του, ένα νέο κόμβο με αναγνωριστικό <tid>. Στη συνέχεια θα πρέπει να διατρέξετε το δένδρο των οπαδών. Για κάθε χρήστη στο δένδρο οπαδών, θα πρέπει να βρείτε το χρήστη στο δένδρο χρηστών. Η αναζήτηση θα γίνεται επαναληπτικά (με χρήση while) καλώντας σε κάθε ανακύκλωση την InorderSuccessor() για να βρείτε τον επόμενο κόμβο του τρέχοντος στην ενδοδιατεταγμένη διάσιχιση. Με αυτό τον τρόπο θα διατρέχετε μία φορά το δένδρο χρηστών προκειμένου να εντοπίσετε όλους τους οπαδούς του χρήστη με αναγνωριστικό <uid>. Αφού εντοπίσετε έναν οπαδό του <uid> στο δένδρο χρηστών, θα εισάγετε στο δένδρο των tweets του, ένα νέο κόμβο με αναγνωριστικό <tid>. Τέλος, θα εισάγετε και μια νέα εγγραφή στον πίνακα κατακερματισμού tweets. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

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

όπου n είναι το πλήθος των κόμβων στο δένδρο των χρηστών, για κάθε i, $1 \leq i \leq n$, m_i είναι το πλήθος των φύλλων του δένδρου 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, αντίστοιχα, που αντιστοιχεί στον j-οστό φύλλο του δένδρου 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 το δένδρο οπαδών του χρήστη με αναγνωριστικό <uid_1>. Κατά το γεγονός αυτό, θα πρέπει να αναζητήσετε το χρήστη με αναγνωριστικό <uid_1> στο δένδρο χρηστών και στη συνέχεια να αφαιρέσετε το χρήστη με αναγνωριστικό <uid_2> από το δένδρο οπαδών του χρήστη με αναγνωριστικό <uid_1>. Έπειτα, θα πρέπει να αναζητήσετε το χρήστη με αναγνωριστικό <uid_2> στο δένδρο των χρηστών, να διατρέξετε το δένδρο των tweets του και να αφαιρέσετε από αυτό τους κόμβους που έχουν το πεδίο uid ίσο με <uid_1> (δηλαδή θα πρέπει να διαγράψετε από το δένδρο 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

όπου n είναι το πλήθος των κόμβων στο δένδρο των χρηστών, για κάθε i, $1 \leq i \leq n$, m_i είναι το πλήθος των φύλλων του δένδρου των tweets του i-οστού $j \in \{1, …, m_i\}$, $tid_{i,j}$ και $uid_{i,j}$, είναι το αναγνωριστικό του tweet και το αναγνωριστικό του χρήστη, αντίστοιχα, που αντιστοιχεί στον j-οστό φύλλο του δένδρου 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 στο δένδρο χρηστών το χρήστη με αναγνωριστικό <uid> και να διαγράψετε όλα τα στοιχεία του δένδρου των tweets αυτού του χρήστη. Στη συνέχεια, θα πρέπει να διατρέξετε το δένδρο των οπαδών του 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 το αναγνωριστικό που αποθηκεύεται στον τρέχοντα κόμβο που διασχίζετε στο δένδρο οπαδών. Τέλος, θα πρέπει να διαγράφετε από το δένδρο χρηστών το χρήστη με αναγνωριστικό <uid>. Επίσης θα πρέπει να διαγράφονται από το πίνακα κατακερματισμού tweets, όλα τα tweets που δημοσιεύτηκαν από το χρήστη <uid> θέτοντας το πεδίο <delete> σε T (TRUE)_ για να φαίνεται ότι το tweet στην συγκεκριμένη θέση διαγράφηκε και είναι διαθέσιμη για την αποθήκευση κάποιου άλλου tweet στο μέλλον. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

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

όπου n είναι το πλήθος των κόμβων στο δένδρο των χρηστών, για κάθε i, $1 \leq i \leq n$, m_i είναι το πλήθος των φύλλων του δένδρου 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, αντίστοιχα, που αντιστοιχεί στον j-οστό φύλλο του δένδρου των tweets του i-οστού χρήστη.

H <uid> <date1> <date2>

Γεγονός τύπου *history tweets* που υποδηλώνει την αναζήτηση των tweets που έχει δημοσιεύσει ο user <uid> μεταξύ των ημερομηνιών (timestamps) <date1> και <date2>. Κατά το γεγονός αυτό θα πρέπει να αναζητείσετε στο δένδρο χρηστών, το χρήστη με

αναγνωριστικό <uid>. Στη συνέχεια θα πρέπει να βρείτε στο δένδρο των tweets του, το φύλλο με το μικρότερο <timestamp> το οποίο είναι ίσο ή μεγαλύτερο από <date1> και χρησιμοποιώντας τους δείκτες <next> των φύλλων να διατρέξετε και να εκτυπώσετε όλα τα tweets που δημοσίευσε ο συγκεκριμένος χρήστης μεταξύ των ημερομηνιών <date1> και <date2> (συμπεριλαμβανομένων των <date1>, <date2>). Η διαδικασία τερματίζει όταν βρείτε κάποιο tweet που έχει <timestamp> μεγαλύτερο από <date2> (ή αν φτάσετε στο τέλος της λίστας των φύλλων). Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

H <uid> <date1> <date2>
User = <tid_1:uid_1:$timestamp_1$>, <tid_2:uid_2:$timestamp_2$> …, <tid_n:uid_n:$timestamp_n$>
DONE

όπου n είναι το πλήθος των φύλλων του δένδρου των tweets του χρήστη με αναγνωριστικό uid όπου βρέθηκαν tweets μεταξύ των δύο συγκεκριμένων ημερομηνιών, και για κάθε $i \in \{1, \ldots, n\}$, tid_i και uid_i είναι το αναγνωριστικό του tweet και το αναγνωριστικό του χρήστη, αντίστοιχα, που αντιστοιχεί στον i-οστό φύλλο του δένδρου των tweets του χρήστη με αναγνωριστικό uid.

L <tid>
Γεγονός τύπου *lookup* που υποδηλώνει την αναζήτηση ενός tweet με αναγνωριστικό <tid> στον πίνακα κατακερματισμού των twets. Κατά το γεγονός αυτό, θα αναζητήσετε στον πίνακα κατακερματισμού των tweets το tweet με αναγνωριστικό <tid>. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

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

όπου tid, uid και timestamp είναι το αναγνωριστικό του tweet, το αναγνωριστικό του χρήστη που δημοσίευσε το tweet και η ημερομηνία στην οποία το tweet δημοσιεύτηκε.

X
Γεγονός τύπου *print users* το οποίο σηματοδοτεί την εκτύπωση όλων των χρηστών που αποθηκεύονται στο δένδρο χρηστών. Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

```
X
    User1 = uid1
     Followers1: <uid1,1>, <uid1,2>, ..., <uid1,n>
     Tweets1: <tid1,1>, <tid1,2>, ..., <tid1,m>
    ...
    Userk = uidk
     Followersk: <uidk,1>, <uidk,2>, ..., <uidk,n>
     Tweetsk: <tidk,1>, <tidk,2>, ..., <tidk,m>
DONE
```

όπου k είναι το πλήθος των κόμβων στο δένδρο των χρηστών, για κάθε i, $1 \leq i \leq k$, n_i είναι το πλήθος των κόμβων στο δένδρο οπαδών του i-οστού χρήστη και για κάθε $j \in \{1, ..., n_i\}$, $uid_{i,j}$ είναι το αναγνωριστικό του j-οστού κόμβου στο δένδρο οπαδών του i-οστού χρήστη (σύμφωνα με την ενδοδιατεταγμένη διάσχιση). Τέλος, m_i είναι το πλήθος των φύλλων του δένδρου των tweets του i-οστού χρήστη και για κάθε $z \in \{1, ..., m_i\}$, $tid_{i,z}$ είναι το αναγνωριστικό του z-οστού φύλλου του δένδρου των tweets του i-οστού χρήστη.

Y

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

```
Y
 Tweets:
        Tweet1 = <tid1:uid1:ts1>
        Tweet2 = <tid1:uid1:ts1>
        ...
        Tweetn = <tidn:uidn:tsn>
DONE
```

όπου n είναι το πλήθος των στοιχείων του πίνακα κατακερματισμού των tweets και για κάθε i, $1 \leq i \leq n$, tid_i, uid_i και ts_i είναι το αναγνωριστικό του tweet, το αναγνωριστικό του χρήστη και το timestamp του tweet που αντιστοιχεί στον i-οστή θέση του πίνακα κατακερματισμού των tweets.

Βαθμολογία Γεγονότων

R	12
S	15
T	15
U	12
D	15
H	15
L	10
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.

```
typedef struct user {
        int uid;
        struct follower *followers;
        struct tweet *wall_head;
        struct user *par;
        struct user *lc;
        struct user *rc;
}user_t;

typedef struct tweet {
        int  tid;
        int  uid;
        int timestamp;
        struct tweet *next;
        struct tweet *lc;
        struct tweet *rc;
}tweet_t;
```

```
typedef struct follower {
        int uid;
        struct follower *lc;
        struct follower *rc;
}follower_t;

typedef struct tweet_hashTable {
        int tid;
        int uid;
        int timestamp;
        char remove;
}t_hashTable;

/* Global variable, pointer to the head of the users tree */
extern user_t *users_p;

* Global variable, pointer to the Guard of the user tree */
extern user_t *users_s;                 // Sentinel

/* Global variable, tweet's hashtable */
extern t_hashTable *tweetHashTable;


/* Global size of tweets hashtable */

extern int max_tweets_g ;


/* Global variable, array of primes */

int primes_g[160];


/* Hash Functions */

#define HFUNC_1(X) ((X) %  (max_tweets_g))

#define HFUNC_2(X) (((X)*(X)) % (max_tweets_g) + 1)
```