HY-486
Αρχές Κατανεμημένου Υπολογισμού
Χειμερινό Εξάμηνο 2020-2021

1η Προγραμματιστική Εργασία

Προθεσμία παράδοσης: Δευτέρα, 30/11 στις 23:59.

1. Γενική Περιγραφή

Στην πρώτη προγραμματιστική εργασία καλείστε να υλοποιήσετε ένα διαμοιραζόμενο σύστημα ανάρτησης ειδήσεων (concurrent blogging system), πάνω στο οποίο θα εκτελούνται ταυτόχρονα διάφορες λειτουργίες.

Η προγραμματιστική εργασία θα πρέπει να υλοποιηθεί με τη γλώσσα C και τη χρήση της βιβλιοθήκης pthreads.

Σημείωση: Η εργασία αποτελείται από φάσεις και είναι, εν μέρει, διαφορετική για τους προπτυχιακούς και τους μεταπτυχιακούς φοιτητές. Κάθε φάση περιγράφεται αναλυτικά παρακάτω.

2. Υλοποίηση

Στην εργασία αυτή θα πρέπει να υλοποιήσετε ένα διαμοιραζόμενο σύστημα ανάρτησης ειδήσεων. Οι λειτουργίες που θα εκτελούνται είναι: i) **ανάρτηση**, ii) **κατηγοριοποίηση** και iii) **έλεγχος** της **εγκυρότητας** των δημοσιεύσεων (για παράδειγμα, εντοπισμός ψευδών ειδήσεων). Θα υπάρχουν τρεις οικογένειες νημάτων (threads):

- **οι εκδότες** εκτελούν την ανάρτηση των ειδήσεων στο σύστημα,
- οι **διαχειριστές** είναι υπεύθυνοι για την κατηγοριοποίηση των αναρτήσεων στην κατάλληλη θεματική ενότητα, για παράδειγμα αστυνομικό περιεχόμενο, lifestyle κλπ, και
- οι **ελεγκτές περιεχομένου** είναι υπεύθυνοι για τον εντοπισμό αναρτήσεων που ενδέχεται να παραβιάζουν τους όρους χρήσης της πλατφόρμας.

Κατά την εκκίνησή του, το πρόγραμμα λαμβάνει ως είσοδο από τη γραμμή εντολών (command line) έναν ακέραιο **N**, ο οποίος καθορίζει το πλήθος των διαχειριστών (**N**), το πλήθος των ελεγκτών περιεχομένου (**N**) και το πλήθος των εκδοτών (**M=2N**). Το συνολικό πλήθος των νημάτων του προγράμματος είναι **M**. Αρχικά, όλα τα **M** νήματα παίζουν το ρόλο των εκδοτών και στη συνέχεια, τα **N** από αυτά θα παίξουν το ρόλο των διαχειριστών και τα υπόλοιπα **N**, θα παίξουν το ρόλο των ελεγκτών περιεχομένου. Για λόγους απλότητας της άσκησης υποθέστε ότι ο **N** είναι άρτιος αριθμός και πολλαπλάσιο του 4.

Α' Φάση

Η πρώτη φάση περιλαμβάνει αρχικά την **ανάρτηση** συγκεκριμένου πλήθους ειδήσεων από τα νήματα εκδότες, με την αποθήκευσή τους σε μια **διαμοιραζόμενη, ταξινομημένη και απλά συνδεδεμένη λίστα,** που λέγεται **λίστα αναρτήσεων**. Έπειτα, μόνο το νήμα εκδότης με αναγνωριστικό **0** θα **επιβεβαιώσει** ότι i) έχουν αναρτηθεί όλες οι ειδήσεις και ii) έχουν σωστά (μοναδικά) αναγνωριστικά.

Η λίστα αναρτήσεων υλοποιείται ως**:**

- **(προπτυχιακοί φοιτητές) Linked List with lazy synchronization**, όπως διδάχθηκε στο μάθημα (Ενότητα 5).
- **(μεταπτυχιακοί φοιτητές) Linked List with optimistic synchronization**, όπως διδάχθηκε στο μάθημα (Ενότητα 5).

Και στις δύο περιπτώσεις, η διαμοιραζόμενη λίστα αναπαρίσταται από το struct **SinglyLinkedList** και κάθε κόμβος της από το struct **LLNode**, τα οποία παρουσιάζονται στη συνέχεια.

```
struct SinglyLinkedList {
    struct LLNode *head;
    struct LLNode *tail;
}
```

```
struct LLNode {
    int postID;
    pthread_mutex_t lock;
    struct LLNode *next;
}
```

Συγκεκριμένα, στο struct **LLNode**, το πεδίο **postID** είναι το μοναδικό αναγνωριστικό της κάθε ανάρτησης, το πεδίο **lock** είναι το κλείδωμα (από τη βιβλιοθήκη των pthreads) που έχει συσχετισθεί με τον κόμβο και το πεδίο **next** είναι δείκτης προς το επόμενο στοιχείο της λίστας. Οπότε, κάθε είδηση περιγράφεται από έναν κόμβο struct **LLNode**.

Κάθε νήμα εκδότης με αναγνωριστικό **j** (**0 ≤ j ≤ M–1**) αναρτά **N** δημοσιεύσεις με μοναδικά αναγνωριστικά (**postID**s): **j+i*M**, όπου **0 ≤ i ≤ N–1**. Επομένως:

- ο εκδότης με αναγνωριστικό **j=0** αναρτά ειδήσεις (struct **LLNode**) με μοναδικά αναγνωριστικά: **0, M, 2*M, 3*M, …, (N-1)*M**
- ο εκδότης με αναγνωριστικό **j=1** αναρτά ειδήσεις με μοναδικά αναγνωριστικά**: 1, 1+M, 1+ 2*M, 1+3*M, …,1+ (N–1)*M**
- …
- ο εκδότης με αναγνωριστικό **j=M-1** αναρτά ειδήσεις με μοναδικά αναγνωριστικά**: M-1, M-1 + M, M-1 + 2*M, …, M-1 + (N–1)*M**

Η ανάρτηση είδησης από τους εκδότες γίνεται με την κλήση της συνάρτηση LL-Insert() της λίστα αναρτήσεων. Επομένως, το πρόγραμμά σας θα πρέπει να δημιουργεί M νήματα, κάθε ένα εκ των οποίων, αρχικά, θα εκτελεί N φορές την LL-Insert().

Πριν ξεκινήσει η επιβεβαίωση της ορθής ανάρτησης των ειδήσεων από το νήμα εκδότης με αναγνωριστικό 0, όλα τα νήματα εκδότες θα πρέπει να έχουν τελειώσει με την ανάρτηση των ειδήσεων που το καθένα εισάγει στη διαμοιραζόμενη λίστα. Για να εξασφαλιστεί αυτό, θα πρέπει να χρησιμοποιήσετε ένα φράγμα συγχρονισμού (barrier) από τη βιβλιοθήκη των pthreads, με όνομα **barrier_1st_phase_end**.

Όταν όλα τα νήματα θα έχουν φτάσει στο φράγμα συγχρονισμού, το νήμα εκδότης με αναγνωριστικό 0 θα πρέπει να πραγματοποιεί τις εξής δύο μετρήσεις για την επιβεβαίωση της ορθής ανάρτησης των ειδήσεων:

1. **Count total list size**: Μέτρηση του συνολικού πλήθους των αναρτήσεων που αποθηκεύτηκαν στη λίστα και τύπωση στην οθόνη του εξής μηνύματος:

   total list size counted (expected: $2N^2$, found: X)

   όπου **$2N^2$** είναι η αναμενόμενη τιμή του συνολικού πλήθους των αναρτήσεων και **X** είναι ο συνολικός αριθμός αναρτήσεων που βρέθηκαν στη λίστα.

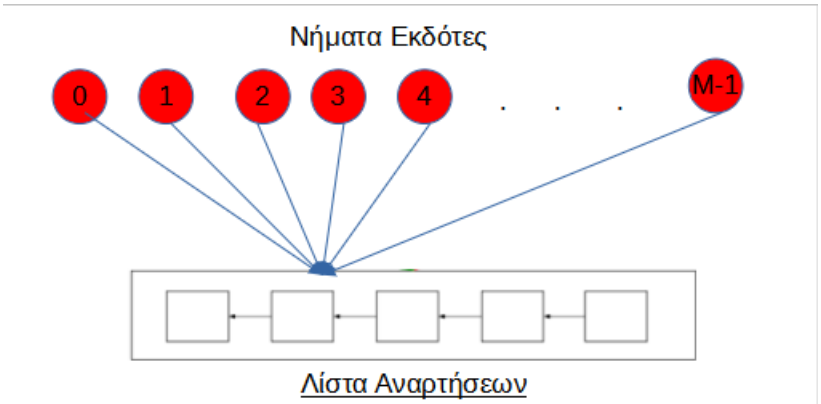
2. **Count total keysum**: Άθροισμα των αναγνωριστικών (**postID**s) των ειδήσεων της λίστας και τύπωση στην οθόνη του εξής μηνύματος:

   total keysum counted (expected:($2N^4$-N^2), found: Y)

   όπου **($2N^4$-N^2)** είναι η αναμενόμενη τιμή του εν λόγω αθροίσματος και **Y** είναι το άθροισμα των αναγνωριστικών των ειδήσεων που βρέθηκαν στη λίστα.

Έπειτα, το νήμα εκδότης με αναγνωριστικό 0 επιβεβαιώνει ότι οι μετρηθείσες τιμές των μετρητών **total list size** και **total keysum** είναι ίδιες με τις αναμενόμενες. Ωστόσο, αν οποιαδήποτε από τις δύο αυτές επιβεβαιώσεις αποτύχει, τότε το πρόγραμμα θα πρέπει να εμφανίζει κατάλληλο μήνυμα λάθους και η εκτέλεσή του να διακόπτεται.

Πριν ξεκινήσει η Β' φάση του προγράμματος, όλα τα νήματα εκδότες θα πρέπει να περιμένουν το νήμα εκδότη με αναγνωριστικό 0 να ολοκληρώσει την επιβεβαίωση της ορθότητας των αναρτήσεων. Για να εξασφαλιστεί αυτό, θα πρέπει να χρησιμοποιήσετε ένα δεύτερο φράγμα συγχρονισμού (barrier) από τη βιβλιοθήκη των pthreads, με όνομα **barrier_2nd_phase_start**.



Σχήμα 1: Απεικόνιση της Α φάσης

Β' Φάση

Στην παρούσα φάση, τα **N** πρώτα νήματα του προγράμματος (δηλαδή αυτά με αναγνωριστικά από **0** έως **N-1**) αναλαμβάνουν το ρόλο των διαχειριστών του συστήματος.

Η δεύτερη φάση περιλαμβάνει αρχικά την **κατηγοριοποίηση** των αναρτήσεων από τα νήματα διαχειριστές, με τη διαγραφή τους από τη λίστα αναρτήσεων και την μεταφορά/αποθήκευσή τους σε ένα σύνολο από **διαμοιραζόμενες ουρές**. Έπειτα, μόνο το νήμα διαχειριστής με αναγνωριστικό **0** θα **επιβεβαιώσει** ότι i) έχουν κατηγοριοποιηθεί όλες οι αναρτήσεις, ii) έχουν σωστά (μοναδικά) αναγνωριστικά και iii) η λίστα αναρτήσεων είναι πλέον κενή. Τα νήματα διαχειριστές θα πρέπει να αναζητήσουν αναρτήσεις (στη διαμοιραζόμενη λίστα) και να τις κατηγοριοποιούν βάσει της θεματολογίας τους, όπως περιγράφεται στη συνέχεια.

Για κάθε κατηγορία αναρτήσεων διατηρούμε μια διαμοιραζόμενη **θεματική ουρά. Το** πλήθος των κατηγοριών (άρα και των θεματικών ουρών) είναι **N/4**. Οι ουρές αποθηκεύονται στον πίνακα **Categories**, ο οποίος έχει **N/4** θέσεις, όπου κάθε θέση αφορά μία θεματική κατηγορία. Κάθε διαμοιραζόμενη ουρά υλοποιείται ως εξής:

- **(Προπτυχιακοί φοιτητές) Unbounded Total Queue με χρήση locks,** όπως διδάχθηκε στο μάθημα (Ενότητα 5, Σελίδες 2 και 3). Η διαμοιραζόμενη ουρά αναπαρίσταται από το struct **queue** και κάθε κόμβος της από το struct **queueNode**, τα οποία παρουσιάζονται στη συνέχεια.

```
struct queue {
    struct queueNode *Head;
    struct queueNode *Tail;
    pthread_mutex_t headLock;
    pthread_mutex_t tailLock;
}
```

```
struct queueNode {
    int postID;
    struct queueNode *next;
}
```

- **(Μεταπτυχιακοί φοιτητές) Unbounded Lock-Free Queue χωρίς τη χρήση locks** (non-blocking algorithm of Michael and Scott), όπως διδάχθηκε στο μάθημα (Ενότητα 5, Σελίδες 4 και 5). Η διαμοιραζόμενη ουρά αναπαρίσταται από το struct **queue** και κάθε κόμβος της από το struct **queueNode**, τα οποία παρουσιάζονται στη συνέχεια.

```
struct queue {
    struct queueNode *Head;
    struct queueNode *Tail;
}
```

```
struct queueNode {
    int postID;
    struct queueNode *next;
}
```

Για κάθε θεματική κατηγορία είναι υπεύθυνα τέσσερα νήματα διαχειριστές. Κάθε νήμα διαχειριστής αρχικά αναζητά (εκτελώντας τη συνάρτηση Search()) στη λίστα αναρτήσεων συγκεκριμένες αναρτήσεις της κατηγορίας για την οποία είναι υπεύθυνο (όπως περιγράφεται στη συνέχεια), και έπειτα, για καθεμία από αυτές τις αναρτήσεις που εντοπίζει i) τη διαγράφει από τη λίστα αναρτήσεων (εκτελώντας τη συνάρτηση LL-Delete()) και ii) την εισάγει στην αντίστοιχη θεματική ουρά (εκτελώντας τη συνάρτηση Enqueue() της ουράς αυτής).

Κάθε νήμα διαχειριστής με αναγνωριστικό **j**, **0 ≤ j ≤ N–1**, εισάγει συνολικά **2N** αναρτήσεις στην ουρά που αποθηκεύεται στη θέση **j mod (N/4)** του πίνακα **Categories**. Για παράδειγμα εάν **N = 8**, τότε το νήμα με αναγνωριστικό **3** θα εισάγει συνολικά **16** αναρτήσεις στην ουρά που αποθηκεύεται στη θέση 1 του πίνακα **Categories** (δηλαδή **Categories[1]**).

Συγκεκριμένα, το νήμα διαχειριστής με αναγνωριστικό **j** εισάγει στην ουρά **Categories[j mod (N/4)]** αναρτήσεις με μοναδικά αναγνωριστικά (**postID**s): **j + i*M** και **j + i*M + N,** όπου **0 ≤ i ≤ N–1.** Επομένως:

- ο διαχειριστής με αναγνωριστικό **j=0** εισάγει αναρτήσεις (**struct queueNode**) με μοναδικά αναγνωριστικά: **0, M, 2*M, 3*M, …, (N-1)*M** και **N, M+N, 2*M + N, 3*M +N , …, (N-1)*M + N**
- ο διαχειριστής με αναγνωριστικό **j=1** εισάγει αναρτήσεις με μοναδικά αναγνωριστικά**: 1, 1+M, 1+2*M, 1+3*M, …, 1+(N–1)*M** και **1 + N, M+ N + 1, 2*M+ N +1, 3*M + N + 1, …, (N–1)*M + N + 1**
- …
- ο διαχειριστής με αναγνωριστικό **j=N-1** εισάγει αναρτήσεις με μοναδικά αναγνωριστικά**: N-1, (N-1) + M, (N-1) + 2*M, …, N-1 + (N–1)*M** και **N-1 + N = 2N-1, M + 2N - 1, 2M + 2N - 1, (N-1)*M + 2N - 1**

Πριν ξεκινήσει η διαδικασία της επιβεβαίωσης από το νήμα διαχειριστή με αναγνωριστικό **0**, όλα τα νήματα διαχειρστές θα πρέπει να έχουν τελειώσει με την κατηγοριοποίηση των αναρτήσεών τους. Για να εξασφαλιστεί αυτό, θα πρέπει να χρησιμοποιήσετε ένα φράγμα συγχρονισμού (barrier) από τη βιβλιοθήκη των pthreads, με όνομα **barrier_2nd_phase_end**.

Κατά τη διαδικασία επιβεβαίωσης, το νήμα διαχειριστής με αναγνωριστικό 0 πραγματοποιεί τις παρακάτω μετρήσεις:

1. **Count total queue sizes:** Μέτρηση του συνολικού πλήθους των αναρτήσεων που αποθηκεύτηκαν στην κάθε θεματική ουρά και τύπωση στην οθόνη των εξής μηνυμάτων:

   Categories[0] queue's *total size counted (expected: 8N, found: X_0)*

   Categories[1] queue's *total size counted (expected: 8N, found: X_1)*

   ...

   Categories[N/4-1] queue's *total size counted (expected: 8N, found: $X_{N/2-1}$)*

   όπου **8N** είναι η αναμενόμενη τιμή του συνολικού πλήθους των αναρτήσεων της κάθε θεματικής ουράς και $\mathbf{X_k}$, **0 ≤ k ≤ N/4–1**, είναι ο συνολικός αριθμός αναρτήσεων που βρέθηκαν στη θεματική ουρά **Categories[k]**.

2. **Count total keysum**: Άρθροισμα των αναγνωριστικών (**postID**s) των αναρτήσεων όλων των θεματικών ουρών και τύπωση στην οθόνη του εξής μηνύματος:

   total keysum check counted(expected: ($2N^4$-N^2), found: Z)

   όπου **($2N^4$-N^2)** είναι η αναμενόμενη τιμή του εν λόγω αθροίσματος και **Z** είναι το άθροισμα των αναγνωριστικών των αναρτήσεων που βρέθηκαν στις θεματικές ουρές.

3. **Count total list's size**: Πλήθος αναρτήσεων που παραμένουν στη λίστα αναρτήσεων και τύπωση

στην οθόνη του εξής μηνύματος:

list's total size counted (expected: 0 , found: L)

όπου **0** είναι το αναμενόμενο πλήθος των αναρτήσεων που θα έπρεπε να έχουν παραμείνει στη λίστα αναρτήσεων και **L** είναι ο συνολικός αριθμός των αναρτήσεων που βρέθηκαν στη λίστα αναρτήσεων.

Έπειτα, το νήμα εκδότης με αναγνωριστικό **0** επιβεβαιώνει ότι οι μετρηθείσες τιμές των μετρητών **queues'** ***total size,*** **queues' total keysum** και **list's total size** είναι ίδιες με τις αναμενόμενες. Ωστόσο, αν οποιαδήποτε από τις επιβεβαιώσεις αυτές αποτύχει, τότε το πρόγραμμα θα πρέπει να εμφανίζει κατάλληλο μήνυμα λάθους και η εκτέλεσή του να διακόπτεται.

Πριν ξεκινήσει η Γ' φάση του προγράμματος, όλα τα υπόλοιπα νήματα (**M-1** σε πλήθος), θα πρέπει να περιμένουν το νήμα διαχειριστή με αναγνωριστικό **0** να ολοκληρώσει την επιβεβαίωση της ορθότητας της κατηγοριοποίησης. Για να εξασφαλιστεί αυτό, θα πρέπει να χρησιμοποιήσετε ένα ακόμη φράγμα συγχρονισμού (barrier) από τη βιβλιοθήκη των pthreads, με όνομα **barrier_3rd_phase_start**. Τα Ν νήματα που δεν παίζουν το ρόλο του διαχειριστή των ειδήσεων, θα πρέπει, μετά το πέρας της $1^{ης}$ φάσης, να περιμένουν απ' ευθείας σ' αυτό το barrier.

Γ' Φάση

Στην παρούσα φάση, τα **N** τελευταία νήματα του προγράμματος (δηλαδή αυτά με αναγνωριστικά από **N** έως **2N-1**) αναλαμβάνουν το ρόλο των ελεγκτών περιεχομένου.

Λόγω πολλαπλών αναφορών από χρήστες του συστήματος για ειδήσεις που παραβιάζουν τους όρους αναρτήσεων, θα πρέπει τα νήματα με ρόλο ελεγκτή περιεχομένου να εξακριβώσουν αν πράγματι πρόκειται για ανακριβείς ειδήσεις. Αν μια ανάρτηση περιέχει ανακριβείς ειδήσεις, θα πρέπει να διαγραφεί προσωρινά, να ελεγχθεί και να διορθωθεί το κείμενο της, έτσι ώστε στη συνέχεια να αναρτηθεί ξανά. Τα πρώτα στάδια της διαδικασίας αυτής πραγματοποιούνται κατά την τρίτη φάση, ενώ η εκ νέου ανάρτηση των διορθωμένων ειδήσεων θα γίνει κατά τη διάρκεια της τέταρτης φάσης.

Έτσι, η τρίτη φάση περιλαμβάνει αρχικά την **ανίχνευση ανακριβών ειδήσεων** από τα νήματα ελεγκτές περιεχομένου. Αυτό προσομοιώνεται ως εξής: Κάθε νήμα ελεγκτής με αναγνωριστικό **j**, **N ≤ j ≤ 2N–1**, διαγράφει (εκτελώντας τη συνάρτηση dequeue()) **N** αναρτήσεις από τη θεματική ουρά **Categories[(j mod N/4)]**.

Στη συνέχεια εισάγει το διορθωμένο κείμενο αυτών των αναρτήσεων σε ένα **δυαδικό νηματικό δένδρο** (εκτελώντας τη συνάρτηση ThreadedBST-Insert()). Ένα παράδειγμα ενός δυαδικού νηματικού δένδρου παρουσιάζεται στο Σχήμα 1. Το δυαδικό νηματικό δένδρο υλοποιείται ως εξής:

- **(προπτυχιακοί φοιτητές) Threaded Binary Search Tree with Fine Grain Synchronization**, όπως αυτό που υλοποιήθηκε στην πρώτη σειρά ασκήσεων.
- **(μεταπτυχιακοί φοιτητές) Threaded Binary Search Tree with Lazy Synchronization**, όπως αυτό που υλοποιήθηκε στην πρώτη σειρά ασκήσεων.

Και στις δύο περιπτώσεις, κάθε κόμβος του δυαδικού νηματικού δένδρου αναπαρίσταται από το struct **treeNode**, το οποίο παρουσιάζεται στη συνέχεια.

```
struct treeNode {
        int postID;
        struct treeNode *lc;
        struct treeNode *rc;
        boolean IsRightThreaded;
        boolean IsLeftThreaded;
        pthread_mutex_t lock;
}
```

Η **ταξινόμηση** των διορθωμένων αναρτήσεων στο δυαδικό νηματικό δένδρο γίνεται βάσει του μοναδικού αναγνωριστικού της ανάρτησης (**postID**). Τα πεδία **lc** και **rc** αποτελούν δείκτες στα

αριστερά και δεξιά παιδιά του δένδρου αντίστοιχα. Οι boolean μεταβλητές **IsRightThreaded** και **IsLeftThreaded** που αποθηκεύονται στο struct του κάθε κόμβου υποδηλώνουν το αν ο κόμβος έχει πραγματικά δεξιό ή αριστερό παιδί (αντίστοιχα) ή αν οι δείκτες **rc** και **lc** είναι νηματικοί, δηλαδή δείχνουν στον διάδοχο (ή προηγούμενο) κόμβο στην ενδοδιατεταγμένη διάσχιση. Το πεδίο **lock** είναι το κλείδωμα (από τη βιβλιοθήκη των pthreads) που έχει συσχετισθεί με τον κόμβο.

Όταν η διαδικασία της διόρθωσης ολοκληρωθεί και το νηματικό δένδρο έχει δημιουργηθεί, το νήμα ελεγκτής με αναγνωριστικό **N** θα **επιβεβαιώσει** ότι έχουν διορθωθεί όλες οι ανακριβείς ειδήσεις, το οποίο σημαίνει ότι i) έχουν αποθηκευτεί όλες στο δυαδικό νηματικό δένδρο και ii) έχουν διαγραφεί οι αντίστοιχες αναρτήσεις από τις θεματικές ουρές.

Πριν ξεκινήσει η διαδικασία της επιβεβαίωσης από το νήμα ελεγκτή περιεχομένου με αναγνωριστικό **N**, όλα τα νήματα ελεγκτές περιεχομένου θα πρέπει να έχουν τελειώσει με τον έλεγχο περιεχομένου των αναρτήσεων. Για να εξασφαλιστεί αυτό, θα πρέπει να χρησιμοποιήσετε ένα φράγμα συγχρονισμού (barrier) από τη βιβλιοθήκη των pthreads, με όνομα **barrier_3rd_phase_end**.

Έτσι, μετά την ολοκλήρωση του ελέγχου περιεχομένου, κατά τη διαδικασία επιβεβαίωσης το νήμα ελεγκτής περιεχομένου με αναγνωριστικό **N** πραγματοποιεί τις παρακάτω μετρήσεις:

1. **Count total tree size:** Μέτρηση του συνολικού πλήθους των αναρτήσεων που αποθηκεύτηκαν δυαδικό νηματικό δένδρο και τύπωση στην οθόνη του εξής μηνύματος:

   tree's total size finished (expected: N^2, found: Y)

   όπου $\mathbf{N^2}$ είναι η αναμενόμενη τιμή του πλήθους των αρτήσεων που θα έπρεπε να υπάρχουν στο δένδρο και **Y** είναι ο αριθμός αναρτήσεων που βρέθηκαν στο δένδρο.

2. **Count total queue sizes**: **:** Μέτρηση του συνολικού πλήθους των αναρτήσεων που υπάρχουν πλέον στην κάθε θεματική ουρά και τύπωση στην οθόνη των εξής μηνυμάτων:

   Categories[0] queue's *total size counted (expected: 4N, found: X_0)*

   Categories[1] queue's *total size counted (expected: 4N, found: X_1)*

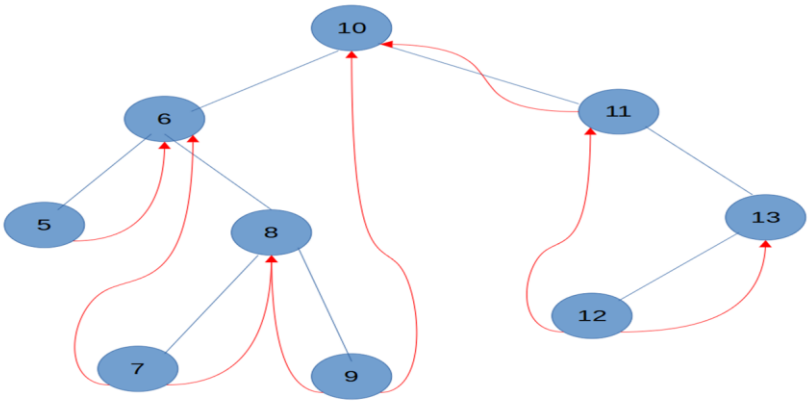
   ...

   Categories[N/4-1] queue's *total size counted (expected: 4N, found: $X_{N/2-1}$)*

   όπου **4N** είναι η αναμενόμενη τιμή του συνολικού πλήθους των αναρτήσεων της κάθε θεματικής ουράς και $\mathbf{X_k}$, $\mathbf{0 \le k \le N/4–1}$, είναι ο συνολικός αριθμός αναρτήσεων που βρέθηκαν στη θεματική ουρά **Categories[k]**.

Έπειτα, το νήμα ελεγκτής περιεχομένου με αναγνωριστικό **N** επιβεβαιώνει ότι οι μετρηθείσες τιμές των μετρητών ***tree's total size*** και **queues'*total size*** είναι ίδιες με τις αναμενόμενες. Ωστόσο, αν οποιαδήποτε από τις επιβεβαιώσεις αυτές αποτύχει, τότε το πρόγραμμα θα πρέπει να εμφανίζει κατάλληλο μήνυμα λάθους και η εκτέλεσή του να διακόπτεται

Πριν ξεκινήσει η Δ' φάση του προγράμματος, όλα τα υπόλοιπα νήματα (**M-1** σε πλήθος), θα πρέπει να περιμένουν το νήμα ελεγκτή περιεχομένου με αναγνωριστικό **N** να ολοκληρώσει την επιβεβαίωση της ορθότητας του ελέγχου περιεχομένου. Για να εξασφαλιστεί αυτό, θα πρέπει να χρησιμοποιήσετε ένα ακόμη φράγμα συγχρονισμού (barrier) από τη βιβλιοθήκη των pthreads, με όνομα **barrier_4th_phase_start**.



Σχήμα 2: Παράδειγμα δυαδικού νηματικού δένδρου. Οι μπλε γραμμές ενώνουν κάθε κόμβο με τα παιδιά του ενώ οι κόκκινες με τον επόμενο ή προηγούμενό του στην ενδοδιατεταγμένη διάταξη. Το δένδρο είναι ταξινομημένο βάσει του πεδίου `postID`

Δ' Φάση

Σε αυτή τη φάση, τα νήματα με αναγνωριστικό από **0** έως **N-1** που έχουν ρόλο διαχειριστή του συστήματος θα πρέπει να αφαιρούν τις διορθωμένες αναρτήσεις από το δυαδικό νηματικό δένδρο και να επανεισάγουν τις διορθωμένες αναρτήσεις πίσω στην κατηγορία ειδήσεων που ανήκει η καθεμία.

Έτσι, η τέταρτη φάση περιλαμβάνει αρχικά την αναζήτηση και τη διαγραφή αναρτήσεων από το δυαδικό νηματικό δένδρο και την επαναεισαγωγή τους στις κατάλληλες θεματικές ουρές από τα νήματα διαχειριστές. Έπειτα, μόνο το νήμα διαχειριστής με αναγνωριστικό **0** θα **επιβεβαιώσει** ότι έχουν επανατοποθετηθεί στις θεματικές ουρές όλες οι διορθωμενες ειδήσεις, το οποίο σημαίνει ότι i) έχει αδειάσει το δυαδικό νηματικό δένδρο, ότι ii) έχουν επανατοποθετηθεί στις θεματικές ουρές όλες οι αναρτήσεις με τα σωστά αναγνωριστικά.

Προκειμένου να γίνει αυτό, το κάθε νήμα διαχειριστής με αναγνωριστικό **j** , **0 ≤ j ≤ N–1** αναζητά στο δυαδικό νηματικό δένδρο (εκτελώντας τη συνάρτηση BST-Search()) αναρτήσεις με αναγνωριστικά της μορφής:

j + i*M καθώς και j + **i*M + N** όπου **0 ≤ i ≤ N–1**

και όσες από αυτές βρει στο δυαδικό νηματικό δένδρο τις διαγράφει (εκτελώντας τη συνάρτηση BST-Delete()) και τις εισάγει στην ουρά με θέση **j mod N/4** του πίνακα **Categories** (όπως ακριβώς έκανε το κάθε ένα από αυτά τα νήματα στη Β φάση).

Πριν ξεκινήσει η διαδικασία της επιβεβαίωσης από το νήμα διαχειριστή με αναγνωριστικό **0**, όλα τα νήματα διαχειριστές θα πρέπει να έχουν τελειώσει τη διαγραφή αναρτήσεων από το δυαδυικό νηματικό δένδρο και την επανατοποθέτησή τους στις θεματικές ουρές. Για να εξασφαλιστεί αυτό, θα πρέπει να χρησιμοποιήσετε ένα φράγμα συγχρονισμού (barrier) από τη βιβλιοθήκη των pthreads, με όνομα **barrier_4th_phase_end**.

Έτσι, μετά την ολοκλήρωση της επανατοποθέτησης όλων των αναρτήσεων στις θεματικες ουρές, κατά τη διαδικασία επιβεβαίωσης το νήμα διαχειριστής με αναγνωριστικό **0** πραγματοποιεί τις παρακάτω μετρήσεις:

1. **Count total tree size:** Μέτρηση του συνολικού πλήθους των αναρτήσεων που αποθηκεύτηκαν δυαδικό νημτικό δένδρο και τύπωση στην οθόνη του εξής μηνύματος:

   tree's total size finished (expected: 0, found: Y)

   όπου **0** είναι η αναμενόμενη τιμή του πλήθους των αναρτήσεων που θα έπρεπε να υπάρχουν στο δένδρο και **Y** είναι ο αριθμός αναρτήσεων που βρέθηκαν στο δένδρο.

2. **Count total queue sizes**: **:** Μέτρηση του συνολικού πλήθους των αναρτήσεων που υπάρχουν πλέον στην κάθε θεματική ουρά και τύπωση στην οθόνη των εξής μηνυμάτων:

   Categories[0] queue's *total size counted (expected: 8N, found: X_0)*

   Categories[1] queue's *total size counted (expected: 8N, found: X_1)*

   ...

   Categories[N/4-1] queue's *total size counted (expected: 8N, found: $X_{N/4-1}$)*

   όπου **8N** είναι η αναμενόμενη τιμή του συνολικού πλήθους των αναρτήσεων της κάθε θεματικής ουράς και $\mathbf{X_k}$, $\mathbf{0 \le k \le N/4-1}$, είναι ο συνολικός αριθμός αναρτήσεων που βρέθηκαν στη θεματική ουρά **Categories[k]**.

3. **Count total keysum**: Άρθροισμα των αναγνωριστικών (**postID**s) των αναρτήσεων όλων των θεματικών ουρών και τύπωση στην οθόνη του εξής μηνύματος:

   total keysum check counted(expected: ($2N^4$-N^2), found: Z)

   όπου **($2N^4$-N^2)** είναι η αναμενόμενη τιμή του εν λόγω αθροίσματος και **Z** είναι το άθροισμα των αναγνωριστικών των αναρτήσεων που βρέθηκαν στις θεματικές ουρές.

Αν οποιαδήποτε από τις επιβεβαιώσεις αυτές αποτύχει, τότε το πρόγραμμα θα πρέπει να εμφανίζει κατάλληλο μήνυμα λάθους και η εκτέλεσή του να διακόπτεται.

3. Αρχικοποίηση και Εκτέλεση Προγράμματος

Το πρόγραμμα θα πρέπει να δέχεται ως είσοδο έναν φυσικό αριθμό **N**, ο οποίος αναπαριστά το πλήθος των νημάτων διαχειριστών και ελεγκτών περιεχομένου. Από αυτόν τον αριθμό θα προκύπτει και ο αριθμός **M = 2N** που ισούται με το πλήθος των νημάτων εκδοτών. Για λόγους ευκολίας μπορείτε να υποθέσετε ότι ο **N** είναι πάντα ζυγός αριθμός και πολλαπλάσιο του 4.

4. Παράδοση Εργασίας

Για την παράδοση της εργασίας θα πρέπει να χρησιμοποιήσετε το πρόγραμμα **turnin**, που υπάρχει εγκατεστημένο στα μηχανήματα του τμήματος. Συγκεκριμένα, η εντολή παράδοσης είναι:

turnin project1@hy486

Για να επιβεβαιώσετε ότι η υποβολή της εργασίας σας ήταν επιτυχής μπορείτε να χρησιμοποιήσετε την εντολή:

verify-turnin project1@hy486

Προσοχή, τα παραδοτέα σας θα πρέπει να περιέχουν ό,τι χρειάζεται και να είναι σωστά δομημένα ώστε να κάνουν compile και να εκτελούνται **στα μηχανήματα της σχολής**, όπου και θα γίνει η εξέταση της εργασίας.

Η προθεσμία παράδοσης της εργασίας είναι την **Δευτέρα 30/11 στις 23:59**.