

Πανεπιστήμιο Ιωαννίνων – Τμήμα Μηχανικών Η/Υ και Πληροφορικής
Δομές Δεδομένων [ΜΥΥ303] – Χειμερινό Εξάμηνο 2021

1^η Εργαστηριακή Άσκηση
Συνδεδεμένες Λίστες

Παράδοση έως Τετάρτη Πέμπτη 3/11, 12:00 από το eCourse

ΠΡΟΣΟΧΗ: Γράψτε σε κάθε αρχείο που παραδίδετε τα ονόματα και τους Α.Μ. των μελών της ομάδας σας. Συμπεριλάβετε όλα τα αρχεία σας (κώδικας Java και lab1results.txt) σε ένα zip αρχείο το οποίο να έχει το όνομα ενός μέλους της ομάδας σας με λατινικούς χαρακτήρες.

Ο σκοπός της άσκησης είναι να υλοποιήσουμε μια **διπλά συνδεδεμένη λίστα** η οποία αποθηκεύει αλφαριθμητικά (αντικείμενα τύπου String). Συγκεκριμένα, κάθε κόμβος της διπλά συνδεδεμένης λίστας, τύπου Node, αποθηκεύει ένα αλφαριθμητικό *str*, το πλήθος των εμφανίσεων του *str* στο αρχείο εισόδου (int count), μια αναφορά στον επόμενο κόμβο της λίστας (Node next), καθώς και μια αναφορά στον προηγούμενο κόμβο της λίστας (Node previous).

Η δομή μας πρέπει να υποστηρίζει τις ακόλουθες βασικές μεθόδους που καλείστε να υλοποιήσετε:

- | | |
|------------------------|--|
| int contains(String s) | Επιστρέφει το πλήθος των εμφανίσεων του αλφαριθμητικού <i>s</i> . Αν το <i>s</i> δεν έχει εισαχθεί στη δομή τότε επιστρέφει μηδέν. |
| void insert(String s) | Εισάγει το αλφαριθμητικό <i>s</i> στη διπλά συνδεδεμένη λίστα. Αν το <i>s</i> υπάρχει ήδη στη δομή, τότε αυξάνουμε κατά ένα το μετρητή των εμφανίσεων του. Στη συνέχεια, τοποθετούμε το <i>s</i> στην κατάλληλη θέση έτσι ώστε τα αλφαριθμητικά να είναι διατεταγμένα σε φθίνουσα σειρά ως προς το πλήθος των εμφανίσεων τους. (Το αλφαριθμητικό με το μέγιστο πλήθος εμφανίσεων θα πρέπει να βρίσκεται στην αρχή της λίστας.) |
| void delete(String s) | Διαγράφει το αλφαριθμητικό <i>s</i> από τη διπλά συνδεδεμένη λίστα. |

Για να ελέγχουμε αν το String *s* είναι αποθηκευμένο σε ένα κόμβο *x*, μπορούμε να γράψουμε *x.str.equals(s) == true*.

Στη συνέχεια υλοποιήστε τις ακόλουθες μεθόδους:

- | | |
|----------------------|---|
| void lexOrder() | Ταξινομεί τη διπλά συνδεδεμένη λίστα σε λεξικογραφική διάταξη (π.χ., το “and” προηγείται του “because”). |
| String print(int k) | Τυπώνει τα <i>k</i> πρώτα αλφαριθμητικά ως προς την τρέχουσα διάταξη της διπλά συνδεδεμένης λίστας, μαζί με το πλήθος των εμφανίσεων τους. |
| String select(int k) | Επιστρέφει το <i>k</i> -οστό αλφαριθμητικό <i>s</i> ως προς την τρέχουσα διάταξη της διπλά συνδεδεμένης λίστας. (Δηλαδή, υπάρχουν <i>k – 1</i> αλφαριθμητικά που προηγούνται του <i>s</i> στη συνδεδεμένη λίστα.) |

**Πανεπιστήμιο Ιωαννίνων – Τμήμα Μηχανικών Η/Υ και Πληροφορικής
Δομές Δεδομένων [ΜΥΥ303] – Χειμερινό Εξάμηνο 2021**

Εκτέλεση Προγράμματος

Η `main()` μέθοδος του προγράμματος `WordList.java` χρησιμοποιεί το πρόγραμμα `In.java` για να διαβάσει από ένα αρχείο εισόδου μία ακολουθία λέξεων, τα οποία εισάγει στη διπλά συνδεδεμένη λίστα. Για την εκτέλεση του προγράμματος θα χρησιμοποιήσουμε για αρχείο εισόδου το κείμενο `TomSawyerB.txt`.

Για την εκτέλεση των προγραμμάτων γράψτε

```
java WordList < TomSawyerB.txt.
```

Το πρόγραμμα αρχικά τυπώνει το χρόνο κατασκευής της διπλά συνδεδεμένης λίστας, αφού ολοκληρωθεί η ανάγνωση του αρχείου εισόδου και τυπώνει το πλήθος των εμφανίσεων των λέξεων `and`, `astonished`, `boat`, `path`, `the`, `train`, `tom` και `wondered`.

Στη συνέχεια, τυπώνει τις 10 πρώτες λέξεις της διπλά συνδεδεμένης λίστας (οι οποίες θα πρέπει να είναι οι 10 πιο συχνές λέξεις του κειμένου). Μετά, διαγράφει με τη σειρά την 9η, 8η και 7η λέξη της διπλά συνδεδεμένης λίστας και τυπώνει τις νέες 10 πρώτες λέξεις.

Τέλος, ταξινομεί τη διπλά συνδεδεμένη λίστα σε λεξικογραφική διάταξη και τυπώνει τις πρώτες 10 λέξεις ως προς τη λεξικογραφική διάταξη.

Αποθηκεύστε τα παραπάνω αποτελέσματα στο αρχείο `lab1results.txt`.

Παραδοτέα

Ανεβάστε στο eCourse ένα zip αρχείο με το τελικό σας πρόγραμμα `WordList.java`, το έτοιμο πρόγραμμα `In.java`, καθώς και το αρχείο των αποτελεσμάτων `lab1results.txt`. Το zip αρχείο πρέπει να έχει το όνομα ενός μέλους της ομάδας σας με λατινικούς χαρακτήρες.