Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής & Υπολογιστών
https://courses.softlab.ntua.gr/progtech/

Προγραμματιστικές Τεχνικές

Άσκηση 10 Δυαδικά Δένδρα

Προθεσμία υποβολής στον grader: 11/5/2018

Υλοποιήστε την κλάση `lexicon` για την αναπαράσταση λεξιλογίων κειμένων με τη μορφή δυαδικών δένδρων αναζήτησης. Η κλάση θα πρέπει να υποστηρίζει τις παρακάτω λειτουργίες, τις οποίες πρέπει να υλοποίησετε:

```
class lexicon {
public:
  lexicon();
  ~lexicon();

  void insert(const string &s);
  int  lookup(const string &s) const;
  int  depth(const string &s);
  void replace(const string &s1, const string &s2);

  friend ostream & operator << (ostream &out, const lexicon &l);
};
```

Θεωρήστε δεδομένο ότι οι λέξεις που θα αποθηκεύονται σε ένα λεξικό δεν θα είναι κενές και θα περιέχουν μόνο μικρά λατινικά γράμματα. Κάθε λέξη θα αντιστοιχεί σε έναν κόμβο στο δένδρο. Ο κόμβος θα περιέχει τη λέξη και τη συχνότητα εμφάνισής της στο λεξικό (δηλαδή έναν μετρητή που θα θυμάται πόσες φορές έχει εισαχθεί αυτή η λέξη). Το αριστερό παιδί του κόμβου θα περιέχει αλφαβητικά (λεξικογραφικά) μικρότερες λέξεις, ενώ το δεξιό παιδί αλφαβητικά (λεξικογραφικά) μεγαλύτερες. Καμία λέξη δεν θα πρέπει να εμφανίζεται σε περισσότερους από έναν κόμβους στο δένδρο. Επίσης, το δένδρο θα πρέπει να είναι ένα απλό BST: δεν απαιτείται και δεν πρέπει να είναι ισοζυγισμένο (π.χ., AVL).

Η μέθοδος `insert(s)` θα εισάγει τη λέξη `s` στο δένδρο.

Η μέθοδος `lookup(s)` θα αναζητά τη λέξη `s` στο δένδρο και θα επιστρέφει τη συχνότητα εμφάνισής. Το αποτέλεσμα θα είναι 0 (μηδέν) αν η λέξη δεν υπάρχει στο δένδρο.

Η μέθοδος `depth(s)` θα αναζητά τη λέξη `s` στο δένδρο, όπως και η `lookup`. Αν δεν υπάρχει, θα επιστρέφει 0 (μηδέν). Αν όμως υπάρχει, θα επιστρέφει το *βάθος* στο οποίο βρίσκεται ο κόμβος που την περιέχει στο δένδρο, δηλαδή το μήκος του μονοπατιού από τη ρίζα έως αυτόν τον κόμβο. Θεωρούμε ότι η ρίζα του δένδρου βρίσκεται σε βάθος 1 (ένα).

Η μέθοδος `replace(s1, s2)` Θα αντικαθιστά όλες τις εμφανίσεις της λέξης `s1` με ισάριθμες εμφανίσεις της λέξης `s2`. Αν η `s1` δεν υπάρχει στο δένδρο, τότε δε θα γίνεται τίποτα. Αν υπάρχει και το πλήθος εμφανίσεών της είναι $k > 0$, τότε η `s1` θα διαγράφεται και θα αναζητάται η λέξη `s2`. Αν αυτή δεν υπάρχει, θα εισάγεται με συχνότητα εμφάνισης k. Αν όμως υπάρχει, τότε η συχνότητα εμφάνισής της θα ενημερώνεται κατάλληλα (θα αυξάνει κατά k).

Κατά τη διαγραφή μίας λέξης από το λεξικό (που γίνεται έμμεσα με τη μέθοδο `replace`), αν διαγράφεται κόμβος που έχει δύο μη κενά παιδιά, τότε ο κόμβος που διαγράφεται αντικαθίσταται από αυτόν

που περιέχει την αμέσως μικρότερη λέξη. Αν διαγράφεται κόμβος που έχει ένα μη κενό παιδί, τότε αντικαθίσταται από το παιδί του.

Η εκτύπωση των λεξικών πρέπει να γίνεται σε αλφαβητική σειρά, με τις λέξεις να ακολουθούνται από τη συχνότητα εμφάνισής τους.

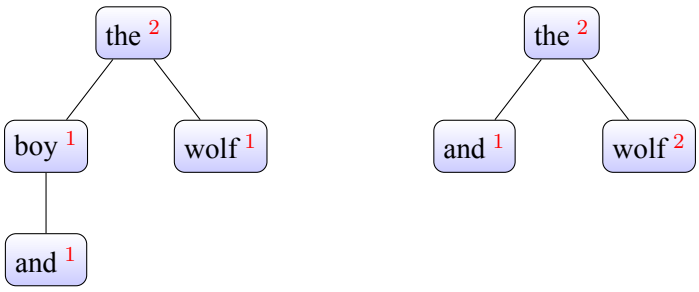
Μπορείτε να δοκιμάσετε την υλοποίησή σας με προγράμματα όπως το εξής:

```
int main() {
  lexicon l;
  l.insert(”the”);
  l.insert(”boy”);
  l.insert(”and”);
  l.insert(”the”);
  l.insert(”wolf”);
  cout << ”The word ’the’ is found ” << l.lookup(”the”) << ” time(s)” << endl;
  cout << ”The word ’and’ is found at depth ” << l.depth(”and”) << endl;
  cout << l;
  l.replace(”boy”, ”wolf”);
  cout << ”After replacement:\n”;
  cout << l;
  cout << ”Now the word ’and’ is found at depth ” << l.depth(”and”) << endl;
}
```

Η εκτέλεσή του θα πρέπει να εμφανίζει:

```
The word ’the’ is found 2 time(s)
The word ’and’ is found at depth 3
and 1
boy 1
the 2
wolf 1
After replacement:
and 1
the 2
wolf 2
Now the word ’and’ is found at depth 2
```

Η μορφή του δένδρου πριν (αριστερά) και μετά (δεξιά) την κλήση της `replace` δίνεται παρακάτω.



Στο αρχείο που θα ανεβάσετε στον grader θα πρέπει να συμπεριλάβετε (μόνο) τη δήλωση της κλάσης `lexicon` και τις υλοποιήσεις των μεθόδων της.

Προσοχή!

Θα προστεθεί μία ακόμη άσκηση, πάνω σε AVL trees, με την ίδια προθεσμία παράδοσης.