Δυαδικό Δένδρο Αναζήτησης (BST)

Υποθέσεις:

- Οι λέξεις με κεφαλαία είναι άλλες από αυτές με πεζά.
- Οι αριθμοί δεν είναι λέξεις.
- Στην διαγραφή στοιχείου διαγράφετε ολόκληρη η θέση.

Μεταβλητές:

Το struct nNode (κόμβος) περιέχει:

- **key,** η λέξη που αποθηκεύεται στον κόμβο.
- **times,** οι φορές που εμφανίζεται η λέξη.
- **right, left,** δείκτες που δείχνουν προς το δεξί και αριστερό παιδί του κόμβου.

Μέθοδοι:

- nNode **CreateLeaf**(string **key**);
  - Ορίσματα :
    Η λέξη **key.**
  - Επιστρέφει:
    Έναν καινούργιο κόμβο **n** .

- void **AddLeaf**(string **key**);
  - Ορίσματα:
    Η λέξη **key.**
  - Επιστρέφει:
    (κενό) καλεί την συνάρτηση **AddLeafPrivate**(string **key**, nNode ***Ptr**).

- void **AddLeafPrivate**(string **key**, nNode ***Ptr**);
  - Ορίσματα:
    Η λέξη **key**
    ¨Ενας δείκτης ***Ptr** που δείχνει σε έναν κόμβο.
  - Επιστρέφει:
    (κενό) καλεί την συνάρτηση **CreateLeaf** αν πρέπει να δημιουργηθεί ένας καινούργιος κόμβος, αλλιώς καλεί αναδρομικά τον εαυτό της για τον επόμενο κόμβο ψάχνοντας να βρει την θέση για την λέξη που της δόθηκε. Αν η λέξη υπάρχει, τότε αυξάνει την μεταβλητή **times** στον κόμβο όπου αυτή είναι αποθηκευμένη.

- void **PrintInOrder**();
    - Ορίσματα:
        (κενό)
    - Επιστρέφει:
        (κενό) καλεί την συνάρτηση **PrintInOrderPrivate**(nNode ***Ptr**).
- void **PrintInOrderPrivate**(**nNode *Ptr**);
    - Ορίσματα:
        ¨Ενας δείκτης ***Ptr** που δείχνει σε έναν κόμβο.
    - Επιστρέφει:
        (κενό) εκτυπώνει στην οθόνη το δένδρο με **inorder** διάσχιση .

- void **PrintPreOrder**();
    - Ορίσματα:
        (κενό)
    - Επιστρέφει:
        (κενό) καλεί την συνάρτηση **PrintPreOrderPrivate**(nNode ***Ptr**).

- void **PrintPreOrderPrivate**(**nNode *Ptr**);
    - Ορίσματα:
        ¨Ενας δείκτης ***Ptr** που δείχνει σε έναν κόμβο.
    - Επιστρέφει:
        (κενό) εκτυπώνει στην οθόνη το δένδρο με **preorder** διάσχιση.

- void **PrintPostOrder**();
    - Ορίσματα:
        (κενό)
    - Επιστρέφει:
        (κενό) καλεί την συνάρτηση **PrintPostOrderPrivate**(nNode ***Ptr**).

- void **PrintPostOrderPrivate**(**nNode *Ptr**);
    - Ορίσματα:
        ¨Ενας δείκτης ***Ptr** που δείχνει σε έναν κόμβο.
    - Επιστρέφει:
        (κενό) εκτυπώνει στην οθόνη το δένδρο με **postorder** διάσχιση .

- Bool **SearchTree**(string **key,** nNode* &**target**);
    - Ορίσματα:
        η λέξη **key** που ψάχνουμε.
        ο δείκτης **target,** ο οποίος καλείται με αναφορά.
    - Επιστρέφει:
        επιστρέφει την τιμή της **SearchTreePrivate.**

- **SearchTreePrivate**(string **key**, nNode ***Ptr**, nNode* **&target**);
    - Ορίσματα:
        η λέξη **key** που ψάχνουμε.
        ¨Ενας δείκτης ***Ptr** που δείχνει σε έναν κόμβο.
        ο δείκτης **target,** ο οποίος καλείται με αναφορά.
    - Επιστρέφει:
        Καλεί τον εαυτό της αναδρομικά με σκοπό να βρει την λέξη **key** σε κάποιον κόμβο. Αν την βρεί επιστρέφει **true**, αν όχι επιστρέφει **false**. Σε περίπτωση που η λέξη βρεθεί, δεδομένα όπως το **key** και το **times** περνούν στον δείκτη **target (**κλήση με αναφορά, ώστε να επιστραφούν τα δεδομένα στην **main)**.

- string **FindSmallest**();
    - Ορίσματα:
        (κενό)
    - Επιστρέφει:
        επιστρέφει την τιμή της **FindSmallestPrivate.**
    -
- string **FindSmallestPrivate**(nNode ***Ptr**);
    - Ορίσματα:
        ¨Ενας δείκτης ***Ptr** που δείχνει σε έναν κόμβο.
    - Επιστρέφει:
        Επιστρέφει το κόμβο που περιέχει την μικρότερη τιμή σε ένα δένδρο. Αν ο αριστερός δείκτης του κόμβου είναι **NULL** (άρα ο κόμβος που δείχνει ο ***Ptr** έχει την μικρότερη τιμή) επιστρέφει τον κόμβο, αλλιώς καλεί τον εαυτό της με όρισμα τον δείκτη στο αριστερό παιδί του κόμβου και συνεχίζει μέχρι να βρει τον κόμβο που περιέχει την μικρότερη τιμή.

- void **RemovenNode**(string **key**);
    - Ορίσματα:
        η λέξη **key** που θέλουμε να αφαιρέσουμε από το δένδρο.
    - Επιστρέφει:
        (κενό) καλεί την συνάρτηση **RemovenNodePrivate**.

- **RemovenNodePrivate**(string **key**, nNode ***Parent**)
    - Ορίσματα:
        η λέξη **key** που θέλουμε να αφαιρέσουμε από το δένδρο.
        ένας δείκτης ***Parent** που δείχνει σε ένα κόμβο που ενδέχεται να είναι ο “γονέας” του κόμβου που περιέχει την λέξη που ψάχνουμε.
    - Επιστρέφει:;
        (κενό) αν η λέξη που πρέπει να αφαιρεθεί είναι η ρίζα (**root**) τότε καλεί την συνάρτηση **RemoveRootMatch**. Διαφορετικά, αν η λέξη **key** είναι μικρότερη από τη λέξη του κόμβου **Parent** και ο αριστερός δείκτης του δεν είναι **NULL**, τότε αν το αριστερό παιδί του **Parent** είναι το **key** καλεί την συνάρτηση **RemoveMatch** για να

την αφαιρέσει, αλλιώς καλεί τον εαυτό της αναδρομικά με όρισμα τον αριστερό δείκτη του **Parent**. Αντίστοιχα, αν η λέξη **key** είναι μεγαλύτερη από τη λέξη του κόμβου **Parent** και ο δεξία δείκτης του δεν είναι **NULL**, τότε αν το δεξί παιδί του **Parent** είναι το **key** καλεί την συνάρτηση **RemoveMatch** για να την αφαιρέσει, αλλιώς καλεί τον εαυτό της αναδρομικά με όρισμα τον δεξί δείκτη του **Parent**. Σε άλλη περίπτωση η λέξη δεν υπάρχει στο δένδρο και τυπώνει αντίστοιχο μήνυμα.

- void **RemoveRootMatch**();
  - Ορίσματα:

    (κενό)
  - Επιστρέφει:

    (κενό) Η **RemoveRootMatch** είναι μια μέθοδος που διαγράφει τον κόμβο της ρίζας (**root)** του δένδρου, δημιουργώντας έναν δείκτη ***delPtr** που δείχνει σε αυτή. Υπάρχουν 3 περιπτώσεις.

    Η πρώτη περίπτωση αφορά ρίζες οι οποίες δεν έχουν κανένα παιδί. Σε αυτή την περίπτωση απλά διαγράφεται η ρίζα (root = NULL) και ο ***delPtr** και το δένδρο παύει να υφίσταται.

    Η δεύτερη περίπτωση αφορά ρίζες οι οποίες έχουν μόνο ένα παιδί. Σε αυτή την περίπτωση η ρίζα αντικαθιστάται με το παιδί της (**root = root->right** ή **root->left**) και διαγράφεται ο **delPtr** (**delPtr->right = NULL** ή **delPtr->left = NULL** και **delete delPtr**).

    Η τρίτη περίπτωση αφορά ρίζες οι οποίες έχουν 2 παιδιά. Σε αυτή την περίπτωση καλείται η βοηθητική μέθοδος **FindSmallestPrivate** με όρισμα τον δεξί δείκτη της ρίζας, ώστε να βρει τον κόμβο που περιέχει την λέξη με μικρότερη τιμή στο δεξί υποδένδρο, η οποία τοποθετείται στην μεταβλητή string **SmallesInRightSubTree**. Στη συνέχεια καλείται η συνάρτηση **RemovenNodePrivate** με όρισμα το **SmallestInRightSubTree** για να αφαιρεθεί αυτή η λέξη από το δένδρο και τελικά να αντικαταστήσει την ρίζα (**root->key = SmallestInRightSubTree**).

- Void **RemoveMatch**(nNode* **parent**, nNode* **match**, bool **left**);
  - Ορίσματα:

    Ένας δείκτης **parent**, ο οποίος δείχνει στον κόμβο-γονέα του κόμβου που βρέθηκε η λέξη που θέλουμε να αφαιρέσουμε.

    Ένας δείκτης **match** που δείχνει στον κόμβο στον οποίο βρέθηκε η λέξη και θέλουμε να διαγράψουμε.

    Μια μεταβλητή τύπου bool **left** που δείχνει αν ο κόμβος **match** είναι αριστερό (true) ή δεξί (false) παιδί του κόμβου **parent.**
  - Επιστρέφει:

    (κενό) Η μέθοδος αυτή είναι παρόμοια με την συνάρτηση **RemoveRootMatch**. Η **RemoveMatch** είναι μια μέθοδος που διαγράφει τον κόμβο **match** του δένδρου, δημιουργώντας έναν δείκτη ***delPtr** που δείχνει σε αυτόν. Υπάρχουν 3 περιπτώσεις.

Η πρώτη περίπτωση αφορά κόμβους οι οποίοι δεν έχουν κανένα παιδί. Σε αυτή την περίπτωση απλά διαγράφεται ο κόμβος και ο ***delPtr** και ανάλογα την τιμή της μεταβλητής **left** προσαρμόζεται ο δείκτης του **parent** (**parent->left = NULL** αν **left == true** ή **parent->right = NULL** αν **left = false**).
Η δεύτερη περίπτωση αφορά κόμβους οι οποίοι έχουν μόνο ένα παιδί. Σε αυτή την περίπτωση ο κόμβος αντικαθιστάται με το παιδί του (αν **left** == true τοτε **parent->left** = **match->right** ή **match->left**, ανάλογα αν το ένα παιδί που έχει είναι αριστερό ή δεξί, διαφορετικά αν **left** == false τοτε παρομοίος **parent->right** = **match->right** ή **match->left**) και στη συνέχεια διαγράφεται ο **delPtr** (**delPtr->right = NULL** ή **delPtr->left = NULL** και **delete delPtr**).
Η τρίτη περίπτωση αφορά κόμβους οι οποίοι έχουν 2 παιδιά. Σε αυτή την περίπτωση καλείται η βοηθητική μέθοδος **FindSmallestPrivate** με όρισμα τον δεξί δείκτη του κόμβου, ώστε να βρει τον κόμβο που περιέχει την λέξη με μικρότερη τιμή στο δεξί υποδένδρο, η οποία τοποθετείται στην μεταβλητή string **SmallesInRightSubTree**. Στη συνέχεια καλείται η συνάρτηση **RemovenNodePrivate** με όρισμα το **SmallestInRightSubTree** για να αφαιρεθεί αυτή η λέξη από το δένδρο και τελικά να αντικαταστήσει τον κόμβο **match** (**match>key = SmallestInRightSubTree**).