Εργασία 3η

Τελική Ημερομηνία Υποβολής - 20.01.2021

Προσοχή: Μην αντιγράφετε τα μηνύματα εξόδου του προγράμματος από την εκφώνηση της εργασίας. Σε αυτές τις περιπτώσεις εισάγονται χαρακτήρες που δεν είναι ASCII, με αποτέλεσμα το script που εκτελείται στο autolab να τερματίζεται αναπάντεχα και χωρίς επαρκή αιτιολόγηση και να μην βγάζει τις πραγματικές διαφορές μεταξύ της δικής σας εξόδου και της επιθυμητής.

Απαιτητές Δομές Δεδομένων της Εργασίας

Για την διεκπεραίωση της παρούσας εργασίας είναι απαραίτητη η υλοποίηση των δομών δεδομένων που περιγράφονται παρακάτω. Οι δομές αυτές αποθηκεύουν ακέραιους τύπου int (ή long int) ή οποιονδήποτε τύπο δεδομένων πιστεύετε ότι σας εξυπηρετεί. Κάθε δομή υλοποιείται από δύο αρχεία, ένα αρχείο με κατάληξη .h και ένα αρχείο με κατάληξη .c. Στα αρχεία με κατάληξη .h περιέχονται τα struct της κάθε δομής δεδομένων και τα prototypes των συναρτήσεων που την υλοποιούν και στο .c μόνο οι υλοποιήσεις των αντίστοιχων συναρτήσεων.

Το δυαδικό δέντρο αναζητήσεως

Στα αρχεία **`tree.c, tree.h`** υλοποιήστε ένα δυαδικό δέντρο αναζητήσεως.

Δυαδικά Δέντρα 1

Σε αυτή την άσκηση θα γράψετε ένα πρόγραμμα το οποίο υλοποιεί την παρακάτω παραλλαγή των δέντρων AVL, η οποία στο εξής θα αποκαλείται `HBTree` (Height Balanced Tree). Ένας κόμβος **K** ενός δυαδικού δέντρου αναζητήσεως `HBTree` θεωρείται ισοζυγισμένος εάν η διαφορά ύψους μεταξύ του αριστερού και του δεξιού υποδέντρου του είναι μικρότερη ή ίση με $\max(1, \lfloor \log_2(N) \rfloor)$, όπου **N** ο αριθμός των κόμβων του υποδέντρου με ρίζα τον κόμβο **K**.

Το δέντρο HBTree εισάγει σε σχέση με το δέντρο AVL ένα επιπλέον πεδίο σε κάθε κόμβο που είναι το βάρος του και αντιπροσωπεύει τον αριθμό των κόμβων που περιεχει το υποδέντρο με ρίζα τον κόμβο αυτό.

Τα είδη των περιστροφών που πραγματοποιούνται για την ισοζύγιση του δέντρου είναι τα ίδια με τα είδη περιστροφών ενός δέντρου AVL. Οι μεθοδολογίες ισοζύγισης του δέντρου κατά την ένθεση και τη διαγραφή του κόμβου είναι όμοιες με τις αντίστοιχες μεθοδολογίες για τα δέντρα AVL. Πιο συγκεκριμένα:

- Κατά την ένθεση ενός νέου κόμβου ανεβαίνουμε από το νέο κόμβο προς τη ρίζα ενημερώνοντας τα ύψη και τα βάρη του κάθε κόμβου στο μονοπάτι. Εάν διαταράσσεται η συνθήκη HBTree του δέντρου σε οποιονδήποτε κόμβο πάνω στο μονοπάτι εφαρμόζουμε το κατάλληλο είδος περιστροφής, Η πράξη είναι τερματική.
- Κατά τη διαγραφή ενός κόμβου ανεβαίνουμε από τον πατέρα του διαγραφέντος κόμβου προς τη ρίζα ενημερώνοντας τα ύψη και τα βάρη του κάθε κόμβου πάνω στο μονοπάτι. Εάν διαταράσσεται η συνθήκη του δέντρου για οποιονδήποτε κόμβο πάνω στο μονοπάτι εφαρμόζουμε το κατάλληλο είδος περιστροφής και εξετάζουμε εάν οι κόμβοι που συμμετείχαν στην περιστροφή είναι ισορροπημένοι με βάση τη συνθήκη HBTree. Εάν κάποιος κόμβος που συμμετείχε στην περιστροφή, μετά από αυτή δεν είναι ισορροπημένος πρέπει να αποκαταστήσουμε αναδρομικά την ισορροπία με κορυφαίο κόμβο τον κόμβο αυτό. Η πράξη της περιστροφής δεν είναι τερματική. Συνεχίζουμε τη διαδικασία μέχρι να φτάσουμε στη ρίζα.

Τα structs και τα prototypes των μεθόδων του δέντρου HBTree θα βρίσκονται στο αρχείο `hbtree.h` και οι υλοποιήσεις των μεθόδων του στο αρχείο `hbtree.c` .

Γράψτε το πρόγραμμα tree1.c το οποίο διαβάζει απο την καθιερωμένη είσοδο μία αναμεμειγμένη ακολουθία εντολών και μην αρνητικών ακεραίων αριθμών. Οι εντολές που μπορούν να διαβαστούν είναι οι εξής:

- **`-i (insert)`**: ένθεση όλων των ακεραίων που ακολουθούν τη συγκεκριμένη εντολή και μέχρι να διαβαστεί νέα εντολή (εκτός της εντολής **`-p`**). Για κάθε αριθμό που επιχειρείται να εισαχθεί στο δέντρο εκτυπώνεται χαρακτήρας αλλαγής γραμμής, το αλφαριθμητικό "`INSERTED X`" εάν η εισαγωγή ήταν επιτυχής ή "`NOT INSERTED X`" εάν η εισαγωγή δεν ήταν επιτυχής, όπου X είναι ο αριθμός που επιχειρήθηκε να εισαχθεί στο δέντρο. Στο τέλος εκτυπώνεται χαρακτήρας αλλαγής γραμμής.
- **`-d (delete)`**: διαγραφή όλων των ακεραίων που ακολουθούν τη συγκεκριμένη εντολή και μέχρι να διαβαστεί νέα εντολή (εκτός της εντολής **`-p`**). Για κάθε αριθμό που επιχειρείται να διαγραφεί από το δέντρο εκτυπώνεται χαρακτήρας αλλαγής γραμμής, το αλφαριθμητικό "`DELETED X`" εάν η διαγραφή ήταν επιτυχής ή "`NOT DELETED X`" εάν η διαγραφή δεν ήταν επιτυχής, όπου X είναι ο αριθμός που επιχειρήθηκε να διαγραφεί από το δέντρο. Στο τέλος εκτυπώνεται χαρακτήρας αλλαγής γραμμής.
- **`-f (find)`**: αναζήτηση όλων των ακεραίων που ακολουθούν τη συγκεκριμένη εντολή στο δέντρο και μέχρι να διαβαστεί νέα εντολή (εκτός της εντολής **`-p`**). Για κάθε ακέραιο που αναζητείται εκτυπώνεται εκτυπώνεται χαρακτήρας αλλαγής γραμμής, το αλφαριθμητικό "`FOUND X`" εάν η αναζήτηση ήταν επιτυχής ή "`NOT FOUND X`" εάν η αναζήτηση δεν ήταν επιτυχής, όπου X είναι ο αριθμός που αναζητήθηκε στο δέντρο. Στο τέλος εκτυπώνεται χαρακτήρας αλλαγής γραμμής.
- **`-p (print)`**: εκτύπωση του δέντρου. Αρχικά εκτυπώνεται χαρακτήρας αλλαγής γραμμής, στη συνέχεια τα περιεχόμενα του δέντρου κατά την pre-order διαπέραση και στο τέλος πάλι χαρακτήρας αλλαγής γραμμής. Για τα περιεχόμενα του δέντρου, μετά από κάθε ακέραιο ακολουθεί κενός χαρακτήρας.
- **`-q (quit)`**: τερματίζει το πρόγραμμα.

  Η εισαγωγή της συγκεκριμένης εντολής δεν αναιρεί την ισχύ της εντολής που προηγήθηκε, όσον αφορά τους αριθμούς που ακολουθούν. Για παράδειγμα, η ακολουθία **`-i 10 5 20 -p 33 48`** αρχικά θα

εισάγει τους αριθμούς 10 5 20, στη συνέχεια θα εκτυπώσει το δέντρο και τέλος θα εισάγει τους αριθμούς 33 48.

Ερώτηση: Ένα δέντρο HBTree έχει λογαριθμικό ύψος; Ποιός είναι ο μέγιστος αριθμός περιστροφών κατά την ένθεση και τη διαγραφή ενός κόμβου του δέντρου;

Δυαδικά Δέντρα 2

Γράψετε το πρόγραμμα `tree2.c`, το οποίο διαβάζει μία ακολουθία θετικών ακεραίων αριθμών από την καθιερωμένη είσοδο (*stdin*), η οποία αποτελεί την pre-order διαπέραση ενός δυαδικού δέντρου αναζητήσεως. Οι αριθμοί της ακολουθίας χωρίζονται μεταξύ τους με ένα ή περισσότερα κενά και η ακολουθία τερματίζει με την ανάγνωση ενός αρνητικού αριθμού. Το πρόγραμμα κατασκευάζει το δυαδικό δέντρο αναζητήσεως που αντιστοιχεί στη συγκεκριμένη διαπέραση.

Στη συνέχεια το πρόγραμμα εξετάζει εάν το δέντρο που διαβάστηκε μπορεί να είναι ένα ερυθρόμαυρο δέντρο. Εάν το δέντρο που διαβάστηκε αντιπροσωπεύει ένα εφικτό ερυθρόμαυρο δέντρο τότε εκτυπώνει το μήνυμα "`RB OK`", ενώ σε διαφορετική περίπτωση εκτυπώνει "`RB NOK`". Ακολουθεί η εκτύπωση δύο χαρακτήρων αλλαγής γραμμής.

Σε περίπτωση που το δέντρο μπορεί να είναι ερυθρόμαυρο, εκτυπώνει την pre-order διαπέραση ως ερυθρόμαυρο δέντρο. Για κάθε κόμβο εκτυπώνεται ο ακέραιος που αντιστοιχεί σε αυτόν, χαρακτήρας '#' (δίεση) και ο χαρακτήρας 'B' αν ο κόμβος είναι μαύρος ή ο χαρακτήρας 'R' αν ο κόμβος είναι κόκκινος. Μετά τη εκτύπωση κάθε κόμβου ακολουθεί κενός χαρακτήρας και μετά το τελευταίο κενό ακολουθεί χαρακτήρας αλλαγής γραμμής.

Κατακερματισμός

Γράψτε ένα πρόγραμμα το οποίο υλοποιεί ένα πίνακα κατακερματισμού με ανοικτή ή ελεύθερη διευθυνσιοδότηση και ~~γραμμική~~ τετραγωνική δοκιμή όσον αφορά τη συνάρτηση τοποθετήσεως. Το πρόγραμμα χρησιμοποιεί ανακατακερματισμό για την διατήρηση του παράγοντα φόρτου (load factor) εντός του επιθυμητού εύρους. Ο πίνακας αποθηκεύει δείκτες σε θετικούς ακέραιους αριθμούς. Μία κενή θέση περιέχει την τιμή ~~0~~ NULL, ενώ μία κενωθείσα (θέση της οποίας το περιεχόμενο έχει διαγραφεί) περιέχει την τιμή (int*)0x1 ~~-1~~.

Ο ανακατακερματισμός δουλεύει ως εξής:

- ~~Μετά~~ Πριν από μία ένθεση εάν ο παράγων φόρτου είναι μεγαλύτερος ή ίσος με ~~από~~ 0.5 ο πίνακας διπλασιάζει το μέγεθος του και τα στοιχεία επανατοποθετούνται σε αυτόν.
- Μετά από μία διαγραφή εάν ο παράγων φόρτου είναι μικρότερος ή ίσος με ~~από~~ 0.125 ο πίνακας μειώνει στο μισό το μέγεθος του και τα στοιχεία επανατοποθετούνται σε αυτόν.

Τα structs και τα prototypes των μεθόδων του πίνακα κατακερματισμού θα βρίσκονται στο αρχείο `htable.h` και οι υλοποιήσεις των μεθόδων του στο αρχείο `htable.c`.

Γράψτε το πρόγραμμα **htable1.c** το οποίο διαβάζει απο την καθιερωμένη είσοδο μία αναμεμειγμένη ακολουθία εντολών και θετικών ακεραίων αριθμών. Οι εντολές που μπορούν να διαβαστούν είναι οι εξής:

- **`-i (insert)`:** ένθεση όλων των ακεραίων που ακολουθούν τη συγκεκριμένη εντολή και μέχρι να διαβαστεί νέα εντολή (εκτός της εντολής **`-p`**). Για κάθε αριθμό που επιχειρείται να εισαχθεί στον πίνακα εκτυπώνεται χαρακτήρας αλλαγής γραμμής, το αλφαριθμητικό "`INSERTED X`" εάν η εισαγωγή ήταν

επιτυχής ή "`NOT INSERTED X`" εάν η εισαγωγή δεν ήταν επιτυχής, όπου X είναι ο αριθμός που επιχειρήθηκε να εισαχθεί. Στο τέλος εκτυπώνεται χαρακτήρας αλλαγής γραμμής.

- **`-d (delete)`**: διαγραφή όλων των ακεραίων που ακολουθούν τη συγκεκριμένη εντολή και μέχρι να διαβαστεί νέα εντολή (εκτός της εντολής **`-p`**). Για κάθε αριθμό που επιχειρείται να διαγραφεί από τον πίνακα εκτυπώνεται χαρακτήρας αλλαγής γραμμής, το αλφαριθμητικό "`DELETED X`" εάν η διαγραφή ήταν επιτυχής ή "`NOT DELETED X`" εάν η διαγραφή δεν ήταν επιτυχής, όπου X είναι ο αριθμός που επιχειρήθηκε να διαγραφεί. Στο τέλος εκτυπώνεται χαρακτήρας αλλαγής γραμμής.
- **`-f (find)`**: αναζήτηση όλων των ακεραίων που ακολουθούν τη συγκεκριμένη εντολή στον πίνακα και μέχρι να διαβαστεί νέα εντολή (εκτός της εντολής **`-p`**). Για κάθε ακέραιο που αναζητείται εκτυπώνεται εκτυπώνεται χαρακτήρας αλλαγής γραμμής, το αλφαριθμητικό "`FOUND X`" εάν η αναζήτηση ήταν επιτυχής ή "`NOT FOUND X`" εάν η αναζήτηση δεν ήταν επιτυχής, όπου X είναι ο αριθμός που αναζητήθηκε. Στο τέλος εκτυπώνεται χαρακτήρας αλλαγής γραμμής.
- **`-p (print)`**: εκτύπωση του πίνακα.
  - Αρχικά εκτυπώνεται χαρακτήρας αλλαγής γραμμής και το αλφαριθμητικό "`SIZE: S, INSERTED: X, DELETED: Y`" ακολουθούμενο από χαρακτήρα αλλαγής γραμμής, όπου S το μέγεθος του πίνακα, X ο αριθμός των κατειλημμένων θέσεων του πίνακα και Y ο αριθμός των θέσεων του πίνακα που έχουν εκκενωθεί και περιέχουν ταφόπλακα.
  - Στη συνέχεια, εκτυπώνονται όλες οι θέσεις του πίνακα ξεκινώντας τη διαπέραση από την πρώτη θέση (θέση 0) έως την τελευταία (θέση S-1) με πλάτος τριών δεκαδικών ψηφίων ακολουθούμενες από κενό χαρακτήρα. Μετά το τελευταίο κενό, εκτυπώνεται χαρακτήρας αλλαγής γραμμής.
  - Τέλος εκτυπώνονται τα περιεχόμενα όλων των ~~κατειλημμένων~~ θέσεων του πίνακα, ξεκινώντας τη διαπέραση από την πρώτη θέση (θέση 0) έως την τελευταία (θέση S-1).
    - Εάν η θέση είναι κατειλημμένη εκτυπώνεται ο ακέραιος που περιέχεται σε αυτή με πλάτος τριών δεκαδικών ψηφίων ακολουθούμενος από κενό χαρακτήρα.
    - Εάν η θέση είναι κενή εκτυπώνονται δύο κενά, αστεράκι '*' και ένα κενό.
    - Εάν η θέση περιέχει ταφόπλακα εκτυπώνονται δύο κενά, δίεση '#' και ένα κενό.
- **`-r (rehashing)`**: Εάν η λειτουργία του ανακατακερματισμού είναι ενεργή, παύει να είναι ενεργή και ο πίνακας παύει να προσαρμόζει στο μέγεθος του με βάση τις τιμές του *load factor*. Εάν η λειτουργία του ανακατακερματισμού είναι μη ενεργή, αυτή ενεργοποιείται.
- **`-q (quit)`**: τερματίζει το πρόγραμμα.

Το πρόγραμμα εκκινεί πάντα από αρχικό μέγεθος πίνακα ίσο με 2 που είναι και το ελάχιστο δυνατό μέγεθος για τον πίνακα. Κατά την εκκίνηση του προγράμματος η λειτουργία ανακατακερματισμού είναι ενεργή.

Τρόπος Αποστολής

Η αποστολή της εργασία θα γίνει μέσω της πλατφόρμας autolab (δεν απαιτείται συνδεση VPN). Ακολουθήστε τα εξής βήματα:

- Φτιάξτε ένα φάκελο με όνομα **`hw3submit`** και αντιγράψτε μέσα εκεί τα αρχεία **hbtree.c, hbtree.h, tree1.c, tree.c, tree.h, tree2.c, htable.h, htable.c, htable1.c**.
- Συμπιέστε ως tar.gz. Σε Linux/KDE πάτε πάνω στο φάκελο, κάνετε δεξί κλικ και επιλέγετε **Compress -> Here (as tar.gz).** Δημιουργείται το αρχείο **hw3submit.tar.gz**.
- Συνδέεστε στη διεύθυνση **https://autolab.e-ce.uth.gr** και επιλέγετε το μάθημα **ECE215-F20 (f20)** και από αυτό την εργασία **HW3**.

- Για να υποβάλετε την εργασία σας κάνετε click στην επιλογή **"I affirm that I have compiled with this course academic integrity policy…"** και πατάτε **submit**. Στη συνέχεια επιλέγετε το αρχείο **hw3submit.tar.g**z που δημιουργήσατε στο βήμα 2.

Παράρτημα - Εκτύπωση δέντρου σε εικόνα

Για την εκτύπωση ενός δέντρου (ή ενός γράφου) σε εικόνα μπορείτε να χρησιμοποιήσετε το πρόγραμμα dot της σουίτας graphviz. Το πρόγραμμα dot μπορεί να σχεδιάσει το δέντρο ή το γράφο σε οποιαδήποτε μορφή εικόνας (png, jpg κλπ) ή σε μορφή pdf εφόσον λάβει ως είσοδο κατάλληλο αρχείο κειμένου που περιέχει την πληροφορία του δέντρου ή του γράφου.

Για κάθε δέντρο που θέλετε ζωγραφίσετε αρκεί να κατασκευάσετε προγραμματιστικά το αντίστοιχο αρχείο κειμένου (ας το πούμε αρχείο dot). Το όνομα και η κατάληξη του αρχείου δεν έχουν σημασία, αλλά συμβατικά ας του δώσουμε την κατάληξη dot.

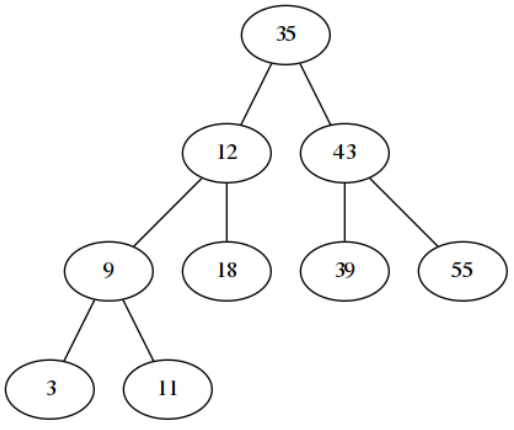
Το περιεχόμενο του αρχείου έχει ως εξής. Κατ' αρχήν, η πληροφορία περικλείεται μέσα σε άγκιστρα όπως παρακάτω:

```
graph G {
  /* εδώ το περιεχόμενο του γράφου ή του δέντρου */
}
```

Η σειρά που εμφανίζονται οι κόμβοι στο αρχείο δεν έχει ιδιαίτερη σημασία. Καλό είναι όμως οποιοδήποτε δέντρο να εκτυπώνεται κατά την pre-order διαπέραση, ώστε η ρίζα να εμφανίζεται στην κορυφή. Κάθε κόμβος έχει ένα μοναδικό αναγνωριστικό, αυτό μπορεί να είναι οτιδήποτε. Για χάριν απλότητας, τοποθετήστε ως μοναδικό αναγνωριστικό κάθε κόμβου τον ακέραιο που έχει αποθηκευμένο στο εσωτερικό του. Τέλος, να θυμάστε ότι δεν έχει ιδιαίτερη σημασία εάν στο αρχείο dot πρώτα εμφανίζονται οι συνδέσεις ενός κόμβου και μετά ο ίδιος ο κομβος.

Ας δούμε το παρακάτω παράδειγμα δέντρου, όπου παρατίθεται το αρχείο dot στα αριστερά και η εικόνα που προκύπτει στα δεξιά. Παρατηρήστε ότι ο συνδέσεις των κόμβων (ακμές του δέντρου) δηλώνονται με δύο κολλητές άνω παύλες (--) μεταξύ των μοναδικών αναγνωριστικών των επιμέρους κόμβων. Τα κενά πριν και μετά τις παύλες είναι προαιρετικά και συμβάλλουν στην αναγνωσιμότητα του αρχείου.

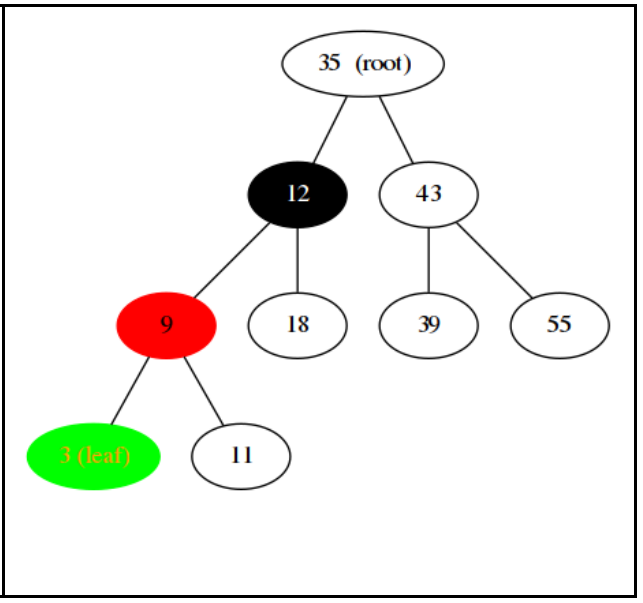
```
graph G {
  35
  35 -- 12
  12
  12 -- 9
  9
  9 -- 3
  3
  9 -- 11
  11
  18
  12 -- 18
  35 -- 43
  43
  43 -- 39
  39
  43 -- 55
  55
}
```



Σε κάθε κόμβο, μπορείτε να έχετε επιπλέον πληροφορία για το πως θέλετε να εμφανίζεται. Η πληροφορία αυτή τοποθετείται αμέσως μετά τον αναγνωριστικό κάθε κόμβου μέσα σε αγκύλες. Εάν δεν προσθέσετε σχετική πληροφορία κάθε κόμβος εμφανίζεται με λευκό κύκλο με μαύρο που περιέχει την πληροφορία στο εσωτερικό του με μαύρα γράμματα (όπως στο προηγούμενο σχήμα).

Παρακάτω ξαναγράφουμε τον αρχείο dot ώστε σε κάποιους κόμβους να αλλάξουμε το χρώμα στο background ή να προσθέσουμε επιπλέον πληροφορία που θέλουμε να εμφανίζεται στον κόμβο ή να αλλάξουμε το αναγνωριστικό τους.

```
graph G {
  theroot [label="35  (root)"]
  theroot -- 12
  12 [style="filled",color="black",fontcolor="white"]
  12 -- 9
  9 [style="filled",color="red",fontcolor="black"]
  9 -- 3
  3 [style="filled",color="green",fontcolor="orange"
label="3 (leaf)"]
  9 -- 11
  11
  18
  12 -- 18
  theroot -- 43
  43
  43 -- 39
  39
  43 -- 55
  55
}
```



Αλλαγές:

1. Η ρίζα του δέντρου άλλαξε μοναδικό αναγνωριστικό. Από 35 έγινε **theroot**. Δεν υπάρχει πρόβλημα εφόσον αυτό να αντικαταστήσει όλες τις εμφανίσεις του 35 ως μοναδικό αναγνωριστικό.
2. Οι κόμβοι 12, 9 και 3 άλλαξαν χρώμα.
3. Στον κόμβο της ρίζας (35) και στον κόμβο 3 αλλάξαμε και το label που εμφανίζεται στην εικόνα.

Από το αρχείο dot στην εικόνα

Εφόσον έχουμε κατασκευάσει σωστά το αρχείο dot ο μετασχηματικός σε εικόνα είναι απλός. Αρκεί να εκτελέσετε από τη γραμμή εντολών την παρακάτω εντολή (με κίτρινο χρώμα).

```
#> dot -T png FILE.dot -o FILE.png
```

όπου FILE.dot το όνομα του αρχείου dot και FILE.png το όνομα του αρχείου που θα προκύψει σε μορφή png. Τα αρχεία dot και png δεν πρέπει να έχουν το ίδιο όνομα, ακολουθούμε όμως “συνεπή” ονοματολογία για να ξέρουμε ποιο αρχείο dot αντιστοιχεί σε κάθε αρχείο png. Εάν δεν σας αρέσει το format png και θέλετε jpg ή pdf μπορείτε να γράψετε.

```
#> dot -T jpg FILE.dot -o FILE.jpg     ή     #> dot -T pdf FILE.dot -o FILE.pdf
```

Προγραμματιστική υλοποίηση του μετασχηματισμού

Αν θέλετε να υλοποιήσετε προγραμματιστικά τον παραπάνω μετασχηματισμό σε C μπορείτε να το κάνετε με τη βοήθεια όσων μάθατε στο μάθημα του Προγραμματισμού II. Αρχικά, δημιουργείστε από τη γονική διεργασία μία νέα διεργασία παιδί. Μέσω της νέας διεργασίας, εκτελέστε το πρόγραμμα dot με τις παραμέτρους που επιθυμείτε. Η γονική διεργασία θα πρέπει να περιμένει τη διεργασία παιδί να ολοκληρώσει την επεξεργασία της πριν συνεχίσει.

Εναλλακτικά μπορείτε να χρησιμοποιήσετε τη συνάρτηση system (man system), που χρησιμοποιεί το shell για την εκτέλεση οποιασδήποτε εντολής.