ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ

Τμήμα Εφαρμοσμένης Πληροφορικής

ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

Εξάμηνο Β'

Φύλλο Ασκήσεων 5: ΔΕΝΤΡΑ

Μάγια Σατρατζέμη, Γεωργία Κολωνιάρη, Αλέξανδρος Καρακασίδης

Παρατηρήσεις:

1. *Τα δεδομένα εισόδου διαβάζονται πάντα με ξεχωριστές εντολές scanf( ) το καθένα (εκτός αν ορίζεται διαφορετικά στην άσκηση) και με τη σειρά που δηλώνονται στις εκφωνήσεις.*
2. *Αντίστοιχα για τα δεδομένα εξόδου και όπου δεν υπάρχουν περαιτέρω διευκρινήσεις για τη μορφή τους, αυτά θα εμφανίζονται με ξεχωριστές εντολές printf( ) το καθένα και με τη σειρά που δηλώνονται στις εκφωνήσεις.*
   - *i) Σε περίπτωση που οι κόμβοι ενός δέντρου περιέχουν περισσότερα από ένα στοιχεία, τότε τα στοιχεία κάθε κόμβου θα εμφανίζονται σε μια γραμμή με ένα κενό χαρακτήρα μεταξύ τους, ενώ κάθε κόμβος θα εμφανίζεται σε διαφορετική γραμμή.*
   - *ii) Αν κατά τη διάσχιση του δέντρου διαπιστώσετε ότι το δέντρο είναι κενό, τότε να εμφανίζετε αντίστοιχα το μήνυμα 'EMPTY TREE'.*
3. *Σε όσες από τις ασκήσεις θεωρείται δεδομένη η ύπαρξη δέντρου θα πρέπει προηγουμένως να το δημιουργήσετε.*
4. **ΠΡΟΣΟΧΗ: Οι ασκήσεις θα πρέπει να λύνονται με χρήση του κώδικα που υλοποιεί τον ΑΤΔ ΔΔΑ. Ο κώδικας που σας δίνεται περιλαμβάνεται στο code.zip στην αντίστοιχη διάλεξη. Οι συναρτήσεις που υλοποιούν τις βασικές λειτουργίες του ΑΤΔ ΔΔΑ δεν τροποποιούνται. Τροποποιήσεις μπορούν να γίνουν ανάλογα με τη άσκηση και εφόσον χρειάζεται μόνο στον τύπο του στοιχείου του ΔΔΑ, και διάσχιση του ΔΔΑ δηλαδή στην εμφάνιση των στοιχείων των κόμβων.**
5. **Στις περιπτώσεις που γίνει τροποποίηση στο BinTreeElementType και το κλειδί δηλωθεί ως αλφαριθμητικό τότε οι εντολές με τη σύγκριση των κλειδιών τροποποιούνται με τη χρήση της συνάρτησης strcmp() και συμπερίληψη της #include <string.h>. Επίσης αν το BinTreeElementType δηλωθεί ως struct και πάλι θα πρέπει να τροποποιήσετε τις εντολές όπου γίνεται σύγκριση των κλειδιών. Οι τροποποιήσεις αφορούν εντολές εντός των συναρτήσεων και όχι στα πρωτότυπα των συναρτήσεων**

---

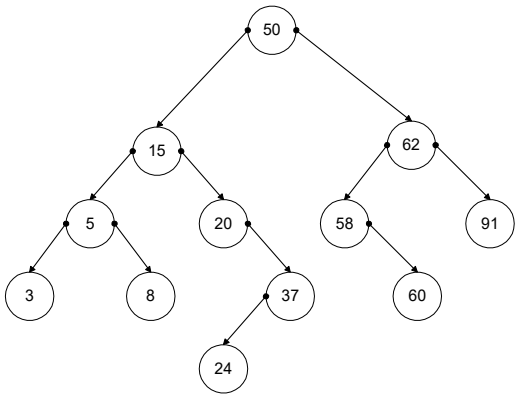
1. Για κάθε μία από τις ακόλουθες λίστες γραμμάτων:
- Σχεδιάστε το ΔΔΑ που προκύπτει όταν τα γράμματα εισάγονται με τη σειρά που δίνονται.
- Αν θεωρήσουμε ότι το ΔΔΑ αποθηκεύεται σε ένα πίνακα, σχεδιάστε τον πίνακα με τα περιεχόμενά του.

a. *A, C, R, E, S*
b. *R, A, C, E, S*
c. *C, A, R, E, S*
d. *S, C, A, R, E*
e. *C, O, R, N, F, L, A, K, E, S*

2. Για κάθε μία από τις ακόλουθες λίστες δεσμευμένων λέξεων της Pascal:
- Σχεδιάστε το ΔΔΑ που προκύπτει όταν οι λέξεις εισάγονται με τη σειρά που δίνονται.
- Εμφανίστε τα περιεχόμενά του παραπάνω ΔΔΑ αν εκτελέσουμε ενδοδιατεταγμένη (inorder), προδιατεταγμένη (preorder), και μεταδιατεταγμένη (postorder) διάσχιση.

a. *program*, *const*, *type*, *function*, *procedure*, *begin*, *end*
b. *array*, *of*, *record*, *case*, *end*, *set*, *file*

3. Θεωρώντας το ακόλουθο ΔΔΑ



a. σχεδιάστε το ΔΔΑ που προκύπτει μετά από την εκτέλεση κάθε μιας από τις παρακάτω λειτουργίες ή αλληλουχίες λειτουργιών:

a1. εισαγωγή του 7

a2. εισαγωγή των 7, 1, 55, 20, και 19

a3. διαγραφή του 8

a4. διαγραφή των 8, 37, και 62

a5. εισαγωγή του 7, διαγραφή του 8, εισαγωγή του 59, διαγραφή του 60, εισαγωγή του 92, και διαγραφή του 50.

b. εμφανίστε τα περιεχόμενά του αρχικού ΔΔΑ αν εκτελέσουμε ενδοδιατεταγμένη (inorder), προδιατεταγμένη (preorder), και μεταδιατεταγμένη (postorder) διάσχιση.

4. Υλοποιήστε την παρακάτω συνάρτηση *GenerateBST* με την οποία θα αποθηκεύονται τυχαία κεφαλαία γράμματα του αγγλικού αλφαβήτου σε ΔΔΑ.

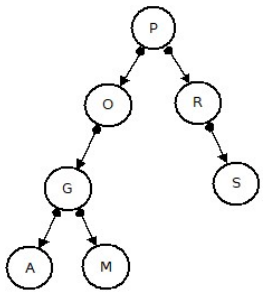
*void GenerateBST(BinTreePointer *Root, int n)*
/ Δέχεται:* *έναν ακέραιο n που υποδηλώνει το πλήθος των τυχαίων γραμμάτων που θα παραχθούν*
Λειτουργία: *παράγει n τυχαία κεφαλαία γράμματα, και εισάγει κάθε ένα από αυτά στο ΔΔΑ*
Επιστρέφει: *το ΓΔΔΑ που δημιουργήθηκε με τη ρίζα του να δεικτοδοτείται από την παράμετρο Root */*

Στο κυρίως πρόγραμμα:

- Θα καλείται η *CreateBST* για τη δημιουργία ενός κενού ΔΔΑ
- Θα εμφανίζεται το μήνυμα '*Give the number of letters?*' και θα διαβάζει έναν ακέραιο αριθμό *n*
- Θα καλείται η *GenerateBST*, η οποία θα δέχεται το κενό ΔΔΑ και τον αριθμό *n*
- Θα καλείται η InorderTraversal για την εμφάνιση του ΔΔΑ που δημιουργήθηκε.
- Το ΔΔΑ περιλαμβάνει όλα τα γράμματα που δημιουργήθηκαν στη διαδικασία *GenerateBST*; *Η απάντηση να δοθεί στον κώδικα μέσα σε σχόλια (χρησιμοποιήστε λατινικούς χαρακτήρες για την απάντηση σας).*

5. Γράψτε μια συνάρτηση *BSTLevel* που επιστρέφει το επίπεδο στο οποίο βρίσκεται ένα στοιχείο σε ένα υπάρχον ΔΔΑ. Η συνάρτηση δέχεται το στοιχείο και το ΔΔΑ και επιστρέφει το επίπεδο ή την τιμή –1 αν το στοιχείο δεν υπάρχει στο ΔΔΑ.
Για να ελέγξετε την ορθότητα της συνάρτησης, στο κυρίως πρόγραμμα:

- Θα καλείται η διαδικασία *CreateBST* για τη δημιουργία ενός κενού ΔΔΑ
- Θα εισάγονται στο δέντρο με τη σειρά οι χαρακτήρες *P, R, O, G, R, A, M, S* οπότε θα προκύψει το διπλανό μη ισοζυγισμένο ΔΔΑ
  - Θα καλείται η συνάρτηση *BSTLevel*, για καθέναν από τους κόμβους του ΔΔΑ και θα εμφανίζεται το μήνυμα:
    Level of x: επίπεδο, όπου *x* η τιμή ενός κόμβου



Ακολουθεί στιγμιότυπο εκτέλεσης:

```
To STOIXEIO EINAI HDH STO DDA

Level of P: 1
Level of R: 2
Level of O: 2
Level of G: 3
Level of A: 4
Level of M: 4
Level of S: 3
```

6. Γράψτε μια συνάρτηση *BSTDepth* που επιστρέφει το βάθος ενός ΔΔΑ.
Για να ελέγξετε την ορθότητα της συνάρτησης, στο κυρίως πρόγραμμα:

- Θα καλείται η διαδικασία *CreateBST* για τη δημιουργία ενός κενού ΔΔΑ

- Θα εισάγονται στο δέντρο με τη σειρά οι χαρακτήρες *P*, *R*, *O*, *C*, *E*, *D*, *U*, *R*, *E*
- Θα καλείται η συνάρτηση *BSTDepth* και θα εμφανίζεται η τιμή που επιστρέφει.

Ακολουθεί στιγμιότυπο εκτέλεσης:

```
TO STOIXEIO EINAI HDH STO DDA
TO STOIXEIO EINAI HDH STO DDA
To vathos toy DDA einai: 5
```

7. Γράψτε πρόγραμμα που θα περιλαμβάνει τις συναρτήσεις *MinBSTValue* και *MaxBSTValue* που επιστρέφουν τη μικρότερη και τη μεγαλύτερη τιμή ενός ΔΔΑ.
Για να ελέγξετε την ορθότητα των συναρτήσεων, στο κυρίως πρόγραμμα:

- Θα καλείται η διαδικασία *CreateBST* για τη δημιουργία ενός κενού ΔΔΑ
- Θα εισάγονται στο δέντρο με τη σειρά οι χαρακτήρες *P*, *R*, *O*, *C*, *E*, *D*, *U*, *R*, *E*, οπότε θα προκύψει το μη ισοζυγισμένο ΔΔΑ της άσκησης 5
- Θα καλούνται οι συναρτήσεις *MinBSTValue* και *MaxBSTValue* και θα εμφανίζονται οι τιμές που επιστρέφουν.

Υπόδειξη: Σε καμία από τις δύο συναρτήσεις δεν χρειάζεται να διασχίσετε όλο το ΔΔΑ. Στην περίπτωση εύρεσης της ελάχιστης τιμής, για παράδειγμα, λόγω των ιδιοτήτων ενός ΔΔΑ μπορείτε απλά να ακολουθήσετε το μονοπάτι που καθορίζουν οι αριστεροί δείκτες μέχρι κάποιος από αυτούς να δείχνει σε κενό υποδέντρο.

8. Γράψτε πρόγραμμα που θα περιλαμβάνει μια συνάρτηση *IdenticalBSTs* που δέχεται δύο ΔΔΑ και επιστρέφει την τιμή *true* ή *false* ανάλογα με το αν έχουν ή όχι ακριβώς την ίδια δομή/περιεχόμενα.
Για να ελέγξετε την ορθότητα της συνάρτησης, στο κυρίως πρόγραμμα:

- Θα καλείται η διαδικασία *CreateBST* για τη δημιουργία τριών κενών ΔΔΑ
- Θα εισάγονται στα δύο πρώτα δέντρα με τη σειρά οι χαρακτήρες *I, D, E, N, T, I, C, A, L* και στο τρίτο οι χαρακτήρες *D, I, F, F, E, R, E, N, T*
- Θα καλείται η συνάρτηση *IdenticalBSTs* για τα δύο πρώτα ΔΔΑ, καθώς επίσης και για τα δύο τελευταία, και θα εμφανίζονται κατάλληλα μηνύματα ανάλογα με την τιμή που επιστρέφει.

Ακολουθεί στιγμιότυπο εκτέλεσης:

```
TO STOIXEIO EINAI HDH STO DDA
TO STOIXEIO EINAI HDH STO DDA
TO STOIXEIO EINAI HDH STO DDA
TO STOIXEIO EINAI HDH STO DDA
Ta dyadika dentra A kai B einai idia
Ta dyadika dentra B kai C diafferoun
```

9. Για κάθε μία από τις ακόλουθες αριθμητικές παραστάσεις, σχεδιάστε ένα δυαδικό δέντρο που αναπαριστά την παράσταση, και έπειτα χρησιμοποιήστε τις κατάλληλες τεχνικές διάσχισης για να βρείτε τις ισοδύναμες προθεματικές και μεταθεματικές εκφράσεις:

a. *(A – B) – C*
b. *A – (B – C)*
c. *A / (B – (C – (D – (E – F))))*
b. *((((A – B) – C) – D) – E) / F*

10. Γράψτε ένα πρόγραμμα για την επεξεργασία ενός ΔΔΑ του οποίου οι κόμβοι περιλαμβάνουν ακέραιους αριθμούς. Ο χρήστης επιτρέπεται να επιλέξει από το ακόλουθο μενού επιλογών:

1.	*Insert element*	(Εισαγωγή στοιχείου)
2.	*Search for an element*	(Αναζήτηση στοιχείου)
3.	*Delete an element*	(Διαγραφή στοιχείου)
4.	*Inorder Traversal*	(Ενδοδιατεταγμένη διάσχιση)
5.	*Preorder Traversal*	(Προδιατεταγμένη διάσχιση)
6.	*Postorder Traversal*	(Μεταδιατεταγμένη διάσχιση)
7.	*Quit*	(Έξοδος)

11. Γράψτε ένα πρόγραμμα που θα πραγματοποιεί ορθογραφικό έλεγχο ενός κειμένου. Οι λέξεις που απαρτίζουν το λεξικό είναι αποθηκευμένες στο αρχείο κειμένου 'I112f5.TXT' (σε κάθε γραμμή του υπάρχει μία λέξη). Διαβάζονται και αποθηκεύονται μία-μία σε ένα ΔΔΑ με τη συνάρτηση *CreateDictionary* και έτσι δημιουργείται το λεξικό-ΔΔΑ. Μετά τη δημιουργία του λεξικού-ΔΔΑ θα εμφανίζετε τις λέξεις του λεξικού. Στη συνέχεια το πρόγραμμα μέσω της συνάρτησης *SpellingCheck* θα διαβάζει ένα κείμενο που είναι αποθηκευμένο στο αρχείο κειμένου 'I111F5.TXT', σε κάθε γραμμή του οποίου υπάρχει μία λέξη και θα διενεργεί ορθογραφικό έλεγχο, δηλαδή θα αναζητά τη λέξη του αρχείου 'I111F5.TXT' στο

λεξικό-ΔΔΑ. Κατά τον ορθογραφικό έλεγχο θα πρέπει να εκτυπώνετε τις λέξεις που δεν βρέθηκαν στο λεξικό και να υπολογίζετε το πλήθος τους. Το πλήθος των λέξεων που δεν βρέθηκαν στο λεξικό θα εμφανίζεται στη main.
Δίνονται τα πρωτότυπα των συναρτήσεων

```
void CreateDictionary(BinTreePointer *Root, FILE *fp);
int SpellingCheck(BinTreePointer Root, FILE *fp);
```

Η 1η παράμετρος της fopen θα πρέπει να περιλαμβάνει το όνομα του αρχείου χωρίς διαδρομή δηλαδή:
fp1 = fopen("i112f5.txt", "r");
fp2 = fopen("i111f5.txt", "r");
Δίνεται στιγμιότυπο εμφάνισης αποτελεσμάτων
**********Dictionary**********
array
begin
boolean
char
const
div
do
else
end
for
function
if
integer
mod
procedure
program
repeat
string
then
type
var
while

**********Not in Dictionary**********
name
Pascal
Turbo
programming
text
pc
Number of words not in Dictionary: 6

12. Σε μια εταιρεία, η μέθοδος με την οποία υπολογίζεται η πληρωμή ενός υπαλλήλου βασίζεται στην κατηγορία του υπαλλήλου: υπάλληλος γραφείου, εργάτης, αντιπρόσωπος. Η εταιρεία διατηρεί ένα αρχείο κειμένου 'I12F5.TXT' με τα στοιχεία των υπαλλήλων. Τα στοιχεία του κάθε υπαλλήλου βρίσκονται σε διαφορετικές γραμμές ως εξής: ονοματεπώνυμο, κωδικός υπαλλήλου):

Ονοματεπώνυμο (επώνυμο όνομα)	*Αλφαριθμητικό 20 χαρακτήρες*
Κωδικός υπαλλήλου	*ακέραιος* (1=υπάλληλος γραφείου, 2=εργάτης, 3=αντιπρόσωπος)

*Γράψτε ένα πρόγραμμα το οποίο θα εκτελεί **με τη σειρά** τις παρακάτω λειτουργίες:*

1. Create BST	Διάβασμα των εγγραφών του αρχείου και δημιουργία ενός ΔΔΑ. Το ΔΔΑ θα δημιουργείται με βάση το ονοματεπώνυμο.
2. Insert employees	Διάβασμα των στοιχείων 3 νέων υπαλλήλων για κάθε μία κατηγορία εργαζομένων και προσθήκη στο ΔΔΑ.
3. Search for an employee by name	Αναζήτηση και εμφάνιση των στοιχείων ενός υπαλλήλου με το ονοματεπώνυμο.
4. List all employees	Εμφάνιση των ονομάτων όλων των υπαλλήλων ταξινομημένων ως προς το ονοματεπώνυμο.

5. List office employees	Εμφάνιση των ονομάτων όλων των υπαλλήλων γραφείου ταξινομημένων ως προς το ονοματεπώνυμο.
6. List factory employees	Εμφάνιση των ονομάτων όλων των εργατών ταξινομημένων ως προς το ονοματεπώνυμο.
7. List sale representatives	Εμφάνιση των ονομάτων όλων των αντιπροσώπων ταξινομημένων ως προς το ονοματεπώνυμο.
8. Delete an employee record	Διαγραφή ενός υπαλλήλου από το ΔΔΑ.

Υπόδειξη: Για τα ερωτήματα 5, 6, 7 τροποποιήστε κατάλληλα τη διαδικασία της ενδοδιατεταγμένης διάσχισης*, δημιουργήστε μια συνάρτηση με βάση την InorderTraversal (δίνεται στο BstADT.c), που θα δέχεται 2 παραμέτρους, τη ρίζα του ΔΔΑ και έναν ακέραιο που θα αντιστοιχεί στο κωδικό του υπαλλήλου.*

```
Give data for office employees:
Give employee name:Zervos Pantelis
Give employee code:1
Give employee name:Iliopoulos Ntinos
Give employee code:1
Give employee name:Georgitsis Fedon
Give employee code:1

Give data for factory employees:
Give employee name:Pantzas Giorgos
Give employee code:2
Give employee name:Karezi Jenny
Give employee code:2
Give employee name:Exarchakos Chronis
Give employee code:2

Give data for sales representatives:
Give employee name:Nathanael Elena
Give employee code:3
Give employee name:Fermas Nikos
Give employee code:3
Give employee name:Vengos Thanasis
Give employee code:3

Give name for employee to lookup:
Vengos Thanasis
Vengos Thanasis, 3

All Employees:
Alexiou Nikos 2
Dimitriou Nikos 1
Exarchakos Chronis 2
Fermas Nikos 3
Georgitsis Fedon 1
Giannou Maria 1
Iliopoulos Ntinos 1
Karezi Jenny 2
Nathanael Elena 3
Nikolaou Marios 3
Pantzas Giorgos 2
Pappas Giannis 2
Ploskas Nikos 2
Satratzemi Maya 3
Totsika Dimitra 2
Vengos Thanasis 3
Zervos Pantelis 1

All office employees:
Dimitriou Nikos 1
Georgitsis Fedon 1
Giannou Maria 1
Iliopoulos Ntinos 1
Zervos Pantelis 1

All factory employees:
Alexiou Nikos 2
Exarchakos Chronis 2
Karezi Jenny 2
Pantzas Giorgos 2
Pappas Giannis 2
Ploskas Nikos 2
Totsika Dimitra 2

All sales representatives:
Fermas Nikos 3
Nathanael Elena 3
Nikolaou Marios 3
Satratzemi Maya 3
Vengos Thanasis 3

Give employee name for deletion:
Zervos Pantelis
```

Συνεχίζεται δίπλα

13. Σε μια εταιρεία, η μέθοδος με την οποία υπολογίζεται η πληρωμή ενός υπαλλήλου βασίζεται στην κατηγορία του υπαλλήλου: υπάλληλος γραφείου, εργάτης, αντιπρόσωπος. Η εταιρεία διατηρεί ένα αρχείο κειμένου 'I13F5TXT' με τα στοιχεία των υπαλλήλων. Τα στοιχεία του κάθε υπαλλήλου βρίσκονται σε διαφορετικές γραμμές Επώνυμο, όνομα, κωδικός κάθε υπαλλήλου):

Επώνυμο	*Αλφαριθμητικό 20 χαρακτήρες*
Ονομα	*Αλφαριθμητικό 20 χαρακτήρες*
Κωδικός υπαλλήλου	*ακέραιος (1=υπάλληλος γραφείου, 2=εργάτης, 3=αντιπρόσωπος)*

Γράψτε ένα πρόγραμμα καθοδηγούμενο από μενού επιλογών για την εκτέλεση των παρακάτω λειτουργιών (κάθε λειτουργία θα υλοποιηθεί ως χωριστή διαδικασία ή συνάρτηση):

1. Create BSTs from file	Διάβασμα των εγγραφών του αρχείου και δημιουργία τριών ΔΔΑ: ένα για τους υπαλλήλους γραφείου, ένα για τους εργάτες και ένα για τους αντιπροσώπους.
2. Insert new employee	Εισαγωγή των στοιχείων ενός υπαλλήλου στο κατάλληλο ΔΔΑ (*παράμετροι*: η εγγραφή με τα στοιχεία του υπαλλήλου και το κατάλληλο ΔΔΑ)
3. Search for an employee	Αναζήτηση και εμφάνιση των στοιχείων ενός υπαλλήλου βάσει επωνύμου (ο χρήστης δίνει το επώνυμο, και τον κωδικό του υπαλλήλου και η αναζήτηση πραγματοποιείται στο αντίστοιχο ΔΔΑ)
4. Print employees	Εμφάνιση των στοιχείων κάθε υπαλλήλου ταξινομημένα ως προς το επώνυμο (ενδοδιατεταγμένη διάσχιση του αντίστοιχου ΔΔΑ). Η εμφάνιση τν στοιχείων κάθε ΔΔΑ θα γίνεται στην ίδια γραμμή και ως εξής: <επώνυμο, όνομα, κωδικός>, <επώνυμο, όνομα, κωδικός>, …
5. Quit	Έξοδος

14. Σε έναν εκπαιδευτικό οργανισμό εργάζονται εκπαιδευτικοί διαφόρων ειδικοτήτων. Τα βασικά τους στοιχεία υπάρχουν σε ένα αρχείου κειμένου 'I14F5.TXT' με την εξής γραμμογράφηση (το αρχείο σας δίνεται).:

Ονοματεπώνυμο (επώνυμο όνομα)	*Αλφαριθμητικό 20 χαρακτήρες*
Τηλέφωνο	*Αλφαριθμητικό 10 χαρακτήρες*
Κωδικός ειδικότητας	*ακέραιος* (1=Θεολόγοι, 2=Φιλόλογοι, …20=Πληροφορικοί)

*Γράψτε ένα πρόγραμμα το οποίο θα εκτελεί **με τη σειρά** τις παρακάτω λειτουργίες* (κάθε λειτουργία θα υλοποιηθεί ως χωριστή διαδικασία ή συνάρτηση):

1. Create BST from file	Διάβασμα των στοιχείων από το αρχείο κειμένου και δημιουργία ενός ΔΔΑ με βάση το <*επώνυμο όνομα*>
2. Print teachers	Εμφάνιση των στοιχείων κάθε εκπαιδευτικού ταξινομημένα ως προς το <επώνυμο όνομα> (ενδοδιατεταγμένη διάσχιση του αντίστοιχου ΔΔΑ). Η εμφάνιση των στοιχείων κάθε εκπαιδευτικού θα γίνεται στην ίδια γραμμή και ως εξής: επώνυμο όνομα, τηλέφωνο, Κωδικός ειδικότητας
3. Insert new teacher	Διάβασμα των στοιχείων δύο νέων εκπαιδευτικών και εισαγωγή των στοιχείων τους στο ΔΔΑ
4. Search for a teacher	Αναζήτηση και εμφάνιση των στοιχείων ενός εκπαιδευτικού βάσει ονοματεπωνύμου *(επώνυμο όνομα)*
5. Search by subject	Αναζήτηση και εμφάνιση κατά αλφαβητική σειρά των στοιχείων των εκπαιδευτικών μιας συγκεκριμένης ειδικότητας (ο κωδικός της ειδικότητας [1..20] αποτελεί παράμετρο της διαδικασίας) ***Υπόδειξη: Τροποποιήστε κατάλληλα τη διαδικασία της ενδοδιατεταγμένης διάσχισης.***
6. Delete a teacher	Διαγραφή ενός εκπαιδευτικού από το ΔΔΑ
7. Print teachers	Όπως το 2

Προσοχή: Θα διαβάζετε το επώνυμο και όνομα (μαζί με το ενδιάμεσο κενό) με μία εντολή και με ξεχωριστές εντολές τηλέφωνο και κωδικό.

Ακολουθεί στιγμιότυπο εκτέλεσης:

```
Print teachers data
dimitriou nikos, 6911111112, 1
giannou maria, 6914441112, 23
nikolaou marios, 6910001112, 15
pappas giannis, 6911111222, 16
satratzemi maya, 6933333332, 20
totsika dimitra, 6911555111, 14

Give teacher full name: georgiou eleni
Give teacher phone number: 6977755588
Give teacher code: 14
```

Give teacher full name: stavrou nikos
Give teacher phone number: 6922222777
Give teacher code: 20

Give teacher full name to search: nikolaou marios
nikolaou marios, 6910001112, 15

Give code to search: 14
georgiou eleni, 6977755588, 14
totsika dimitra, 6911555111, 14

Give teacher full name to delete: pappas giannis

Print teachers data
dimitriou nikos, 6911111112, 1
georgiou eleni, 6977755588, 14
giannou maria, 6914441112, 23
nikolaou marios, 6910001112, 15
satratzemi maya, 6933333332, 20
stavrou nikos, 6922222777, 20
totsika dimitra, 6911555111, 14

15. Υλοποιήστε την παρακάτω συνάρτηση: int RightNodeCount(BinTreePointer Root); η οποία δέχεται σαν είσοδο τη ρίζα ενός ΔΔΑ (Root) και επιστρέφει το πλήθος των δεξιών κόμβων-παιδιών του ΔΔΑ. Δημιουργήστε ένα ΔΔΑ με ακέραιους αριθμός. Το πρόγραμμα θα διαβάζει πρώτα το πλήθος των στοιχείων προς εισαγωγή στο ΔΔΑ, και μετά 1-1 τα στοιχεία αυτά. Στη συνέχεια, θα εμφανίζονται τα στοιχεία του δέντρου σε αύξουσα διάταξη, και τέλος θα καλείται η συνάρτηση RightNodeCount και θα εμφανίζεται το αποτέλεσμα της. Ακολουθεί στιγμιότυπο εκτέλεσης.

Enter number of elements for the tree: 7
Enter number: 20
Enter number: 10
Enter number: 12
Enter number: 25
Enter number: 22
Enter number: 30
Enter number: 40
Elements of BST in increasing order
10 12 20 22 25 30 40
RightNodeCount = 4

16. Εφαρμόστε τον αλγόριθμο του **Huffman** για την κωδικοποίηση των χαρακτήρων που φαίνονται στον παρακάτω πίνακα:

Χαρακτήρας	**Βάρος**
A	0.2
B	0.1
C	0.08
D	0.08
E	0.40
F	0.05
G	0.05
H	0.04

Χρησιμοποιώντας την κωδικοποίηση των χαρακτήρων που προέκυψε από την εφαρμογή του αλγορίθμου Huffman, κωδικοποιήστε το ακόλουθο μήνυμα: “feed a deaf aged hag”.

17. Η κωδικοποίηση **Huffman** για τις παρακάτω δεσμευμένες λέξεις της Pascal που φαίνονται στον πίνακα υπάρχει στο αρχείο ‘codesRW.txt’, ενώ στο αρχείο ‘program.txt’ υπάρχει κωδικοποιημένο ένα πρόχειρο πρόγραμμα. Το αρχείο ‘codesRW.txt’ έχει σε διαφορετικές γραμμές τις δεσμευμένες λέξεις και την κωδικοποίησή τους (πρώτα η δεσμευμένη λέξη και στην επόμενη γραμμή η κωδικοποίησή της). Να γράψετε πρόγραμμα που θα αποκωδικοποιεί και θα εμφανίζει το περιεχόμενο του αρχείου ‘program.txt’.

Δεσμευμένη λέξη	Κωδικοποίηση
begin	10
end	11
for	000
if	01
while	001

18. Χρησιμοποιώντας την κωδικοποίηση **Huffman** που φαίνεται παρακάτω για τους χαρακτήρες A, B, C, D, E αποκωδικοποιήστε τα ακόλουθα αλφαριθμητικά:

Χαρακτήρας	Κωδικοποίηση
A	01
B	0000
C	0001
D	001
E	1

a. 000001001
b. 001101001
c. 000101001
d. 00001010011001

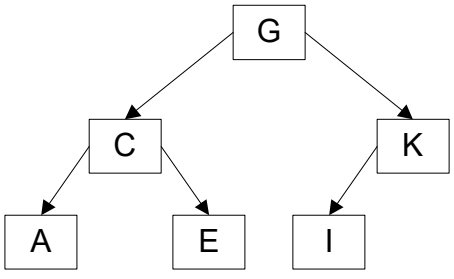
19. Σχεδιάστε το **Β-δέντρο** τάξης 3 που προκύπτει αν εισαχθούν τα ακόλουθα κλειδιά με τη σειρά που δίνονται: *C*, *O*, *R*, *N*, *F*, *L*, *A*, *K*, *E*, *S*.

20. Σχεδιάστε το **Β-δέντρο** τάξης 5 που προκύπτει αν εισαχθούν οι παρακάτω ακέραιοι με τη σειρά που δίνονται: 261, 381, 385, 295, 134, 400, 95, 150, 477, 291, 414, 240, 456, 80, 25, 474, 493, 467, 349, 180, 370, 257.

21. Ανιχνεύστε τη δημιουργία του δέντρου **AVL** που προκύπτει από την εισαγωγή των παρακάτω δεσμευμένων λέξεων της Pascal με τη συγκεκριμένη σειρά:
div, *mod*, *not*, *and*, *or*, *in*, *nil*
Σχεδιάστε το δέντρο και τους παράγοντες ισοζύγισης για κάθε κόμβο πριν και μετά από κάθε αναδιάταξη.

22. Ανιχνεύστε τη δημιουργία του δέντρου **AVL** που προκύπτει από την εισαγωγή των παρακάτω αριθμών με τη συγκεκριμένη σειρά:
22, 44, 88, 55, 11, 99, 77, 33
Σχεδιάστε το δέντρο και τους παράγοντες ισοζύγισης για κάθε κόμβο πριν και μετά από κάθε αναδιάταξη.

23. Θεωρώντας δεδομένο το παρακάτω δέντρο **AVL**, δείξτε τι αποτέλεσμα θα έχει κάθε μια από τις παρακάτω ενέργειες. Σχεδιάστε τις αλλαγές στη δομή του δέντρου και σημειώστε τους παράγοντες ισοζύγισης.



a. Πρόσθεση ενός κόμβου με τιμή κλειδιού D.
b. Πρόσθεση ενός κόμβου με τιμή κλειδιού H.
c. Πρόσθεση ενός κόμβου με τιμή κλειδιού J.
d. Διαγραφή του κόμβου I.
e. Διαγραφή του κόμβου E.
f. Διαγραφή του κόμβου G.

24. Υλοποιείστε την ενδοδιατεταγμένη διάσχιση (Inorder traversal) με μη αναδρομική συνάρτηση. Στο κυρίως πρόγραμμα για να ελεγχθεί η ορθότητα της συνάρτησης, θα δημιουργείται ένα ΔΔΑ με ακέραιους αριθμός που θα δίνει ο χρήστης. Η εισαγωγή των στοιχείων θα σταματάει αν δοθεί ο αριθμός -1. Στη συνέχεια, θα καλείται πρώτα η αναδρομική ενδοδιατεταγμένη διάσχιση και στη συνέχεια η δική σας μη αναδρομική συνάρτηση ενδοδιατεταγμένης διάσχισης. (*Βοήθεια:δείτε στο αρχείο http://eclass.uoa.gr/modules/document/file.php/D419/PDFs%202015-16/12a%20BW%20Trees.pdf τις σελίδες 43-44*).
Ακολουθεί στιγμιότυπο εκτέλεσης.

```
Enter numbers to insert into tree (-1 to stop)
Enter number: 5
Enter number: 8
Enter number: 2
Enter number: 12
Enter number: 25
Enter number: 3
Enter number: 11
Enter number: 7
Enter number: -1

Recursive Inorder Traversal
2 3 5 7 8 11 12 25
Non Recursive Inorder Traversal
2 3 5 7 8 11 12 25
```

25. Υλοποιείστε την αναδρομική συνάρτηση `int CountLeaves(BinTreePointer Root)` η οποία θα υπολογίζει το πλήθος των φύλλων ενός ΔΔΑ το οποίο και θα επιστρέφει. Στο ΔΔΑ θα καταχωρούνται ακέραιοι. Θα εμφανίζονται οι κόμβοι του ΔΔΑ σε αύξουσα διάταξη και στη συνέχεια θα εμφανίζεται το πλήθος των φύλλων του ΔΔΑ στο κυρίως πρόγραμμα. Δίνεται ένα στιγμιότυπο εκτέλεσης:

```
Enter a number for insertion in the Tree: 30
Continue Y/N: Y
Enter a number for insertion in the Tree: 22
Continue Y/N: Y
Enter a number for insertion in the Tree: 50
Continue Y/N: Y
Enter a number for insertion in the Tree: 10
Continue Y/N: Y
Enter a number for insertion in the Tree: 25
Continue Y/N: Y
Enter a number for insertion in the Tree: 55
Continue Y/N: Y
Enter a number for insertion in the Tree: 5
Continue Y/N: Y
Enter a number for insertion in the Tree: 12
Continue Y/N: Y
Enter a number for insertion in the Tree: 52
Continue Y/N: N
Elements of BST
5 10 12 22 25 30 50 52 55
Number of leaves 4
```

26. Δίνεται το αρχείο φοιτητών “foitites.dat”. Κάθε στοιχείο του αρχείου αυτού είναι μια εγγραφή με πεδία:
Αριθμός μητρώου (ΑΜ: int), Επώνυμο (αλφαριθμητικό 20 χαρακτήρες), Ονομα (αλφαριθμητικό 20 χαρακτήρες), Φύλο (χαρακτήρας, τιμή F/M), Ετος (int), Βαθμός (μέσος όρος: float). Για την πιο αποτελεσματική επεξεργασία του αρχείου αυτού δημιουργείται ένα ευρετήριο (index) ως ΔΔΑ. Κάθε στοιχείο του ευρετηρίου αυτού αποτελείται από τον ΑΜ και τον αντίστοιχο αριθμό εγγραφής (γραμμής) στο αρχείο “foitites.dat”. Η αρίθμηση των γραμμών του αρχείου ξεκινούν από την τιμή 0.
Γράψτε ένα πρόγραμμα που εκτελεί τις παρακάτω λειτουργίες και χρησιμοποιεί ως ευρετήριο ένα ΔΔΑ:

1. Δημιουργία του index (ΔΔΑ) από το αρχείο “foitites.dat”.
   Θα διαβάζει 1-1 τις εγγραφές του αρχείου “foitites.dat”και θα καταχωρεί στο ΔΔΑ τον ΑΜ του φοιτητή και τον αντίστοιχο αύξοντα αριθμό εγγραφής (γραμμής) στο αρχείο. Θα επιστρέφει το πλήθος των κόμβων του ΔΔΑ.
   Συνάρτηση `int BuildBST(BinTreePointer *Root);`

2. Θα εμφανίζει το πλήθος των κόμβων του ΔΔΑ όπως και τους κόμβους του ΔΔΑ με αύξουσα διάταξη ως προς ΑΜ.
3. Εισαγωγή νέων εγγραφών φοιτητών στο αρχείο foitites.dat και ενημέρωση του ΔΔΑ. Κάθε αλφαριθμητικό πεδίο όπως και το πεδίο φύλο (τύπου χαρακτήρας) να διαβάζονται με scanf() και στη συνέχεια getchar(). Για το φύλο δε χρειάζεται να γίνεται έλεγχος εγκυρότητας θεωρούμε ότι θα δοθεί F ή M. Η fopen θα κληθεί με 2η παράμετρο “a” καθώς οι νέες εγγραφές θα προστεθούν μετά την τελευταία εγγραφή του αρχείου. Μετά από κάθε προσθήκη εγγραφής στο αρχείο θα εμφανίζετε το μέγεθος του αρχείου. (δείτε στο στιγμιότυπο εκτέλεσης).
   Συνάρτηση `void writeNewStudents(BinTreePointer *Root, int *size);`
4. Θα εμφανίζει το πλήθος των κόμβων του ΔΔΑ όπως και τους κόμβους του ΔΔΑ με αύξουσα διάταξη ως προς ΑΜ.
5. Αναζήτηση φοιτητή. Θα δίνεται ο ΑΜ του φοιτητή και θα τον αναζητά στο ΔΔΑ. Στη συνέχεια εφόσον υπάρχει στο ΔΔΑ θα τον εντοπίζει στο αρχείο “foitites.dat” και θα εμφανίζει όλες τις πληροφορίες της αντίστοιχης εγγραφής. Αν δεν υπάρχει στο ΔΔΑ θα εμφανίζει σχετικό μήνυμα.
6. Θα εμφανίζει το πλήθος των κόμβων του ΔΔΑ όπως και τους κόμβους του ΔΔΑ με αύξουσα διάταξη ως προς ΑΜ.
7. Εκτύπωση των στοιχείων όλων των φοιτητών που είναι καταχωρημένοι στο αρχείο “foitites.dat” με ΜΟ μεγαλύτερο από ένα δοσμένο βαθμό (πχ 0).
   Συνάρτηση `void printStudentsWithGrade(float theGrade);`
   Το αρχείο “foitites.dat” θα «ανοίγει» και θα «κλείνει» σε κάθε συνάρτηση που χρησιμοποιείται και με την κατάλληλη παράμετρο (“a” για εγγραφή στο τέλος του αρχείου ή “r” για διάβασμα των εγγραφών του αρχείου).
   Για κάθε μια από τις παραπάνω λειτουργίες εμφανίζεται στη main() σχετικό μήνυμα (Qx…)
   Δίνεται ένα στιγμιότυπο εκτέλεσης όπου φαίνονται και τα σχετικά μηνύματα .

```
Q1. Build BST from a file
Q2. Print size and BST
Size=8
Nodes of BST
(5, 3), (8, 1), (10, 4), (11, 6), (12, 0), (23, 2), (30, 7), (32, 5),
Q3. Write new students, update file and BST
Give student's AM? 2
Give student surname? DIMITRIOU
Give student name? DIMITRIS
Give student sex F/M? M
Give student's registration year? 2020
Give student's grade? 5
Size=9

Continue Y/N: Y
Give student's AM? 15
Give student surname? ANASTASIOU
Give student name? TASA
Give student sex F/M? F
Give student's registration year? 2019
Give student's grade? 4
Size=10

Continue Y/N: N
Q4. Print size and BST
Size=10
Nodes of BST
(2, 8), (5, 3), (8, 1), (10, 4), (11, 6), (12, 0), (15, 9), (23, 2), (30, 7), (32, 5),
Q5. Search for a student
Give student's code? 2
2, DIMITRIS, DIMITRIOU, M, 2020, 5.0
Q6. Print size and BST
Size=10
Nodes of BST
(2, 8), (5, 3), (8, 1), (10, 4), (11, 6), (12, 0), (15, 9), (23, 2), (30, 7), (32, 5),
Q7. Print students with grades >= given grade
Give the grade: 0
12, KWSTAS, PAPANIKOLAOU, M, 2015, 8.1
8, MICHALIS, ANTONIOU, M, 2014, 5.3
23, RALLIA, RALLIDOU, F, 2016, 6.2
5, ZINA, ZINIDOU, F, 2013, 7.4
```

10, KWSTAS, KWSTIDIS, M, 2010, 6.1
32, ANTONIS, ANTWNIOU, M, 2011, 7.0
11, ANNA, ANNANIDOU, F, 2016, 8.0
30, ALKINOOS, ALKINIDIS, M, 2009, 5.4
2, DIMITRIS, DIMITRIOU, M, 2020, 5.0
15, TASA, ANASTASIOU, F, 2019, 4.0

27. Τα δέντρα radix αποτελούν μία κατηγορία δέντρων, όπου τα παιδιά τα οποία δεν έχουν αδέρφια (κόμβους με τον ίδιο γονέα) συγχωνεύονται με τον κόμβο του πατέρα. Γράψτε συνάρτηση η οποία δέχεται ένα δέντρο με τύπο δεδομένων char[50] και το μετατρέπει σε radix. Στη συνέχεια γράψτε κυρίως πρόγραμμα το οποίο διαβάζει 4 λέξεις, τις αποθηκεύει σε ένα δυαδικό δέντρο και στη συνέχεια το μετατρέπει σε radix.

```
Give the word 1: b
Give the word 2: ba
Give the word 3: bab
Give the word 4: abb

--Inorder BST:
abb b ba bab
--Inorder Radix:
abb b babab
```

28. Γράψτε 2 συναρτήσεις οι οποίες θα βρίσκουν το «πάτωμα» και το «ταβάνι» ενός αριθμού σε ένα Δυαδικό Δένδρο Αναζήτησης (ΔΔΑ), δηλαδή το αμέσως μικρότερο και το αμέσως μεγαλύτερο στοιχείο σε σχέση με τον αριθμό. Σε περίπτωση που ο αριθμός υπάρχει στο ΔΔΑ, τότε το πάτωμα και το ταβάνι είναι ο ίδιος ο αριθμός. Στη συνέχεια, γράψτε κυρίως πρόγραμμα το οποίο διαβάζει θετικούς αριθμούς και τους εισάγει στο ΔΔΑ μέχρι να δοθεί αρνητικός αριθμός, το πρόγραμμα ζητά από τον χρήστη αριθμούς και του επιστρέφει το πάτωμα και το ταβάνι, έως ότου εισαχθεί αρνητικός αριθμός.

```
Enter number to insert: 20
Enter number to insert: 10
Enter number to insert: 30
Enter number to insert: 5
Enter number to insert: 15
Enter number to insert: -1
Enter number to search: 5
Ceiling : 5
Floor: 5
Enter number to search: -1
```

```
Enter number to insert: 20
Enter number to insert: 10
Enter number to insert: 30
Enter number to insert: 5
Enter number to insert: 15
Enter number to insert: -1
Enter number to search: 4
Ceiling : 5
Floor: -1
Enter number to search: -1
```

```
Enter number to insert: 20
Enter number to insert: 10
Enter number to insert: 30
Enter number to insert: 5
Enter number to insert: 15
Enter number to insert: -1
Enter number to search: 17
Ceiling : 20
Floor: 15
Enter number to search: -1
```

29. Γράψτε πρόγραμμα που θα δέχεται για κάθε άτομο τον ΑΜΚΑ (ακέραιος), τον ΑΦΜ (ακέραιος), την ηλικία (ακέραιος). Θα καταχωρεί τα στοιχεία του κάθε ατόμου σε 2 καταλόγους ανάλογα με την ηλικία του, αυτούς με ηλικία μικρότερη ή ίση των 60 ετών και αυτούς με ηλικία μεγαλύτερη των 60.
Κάθε κατάλογος θα πρέπει να οργανωθεί ως ΔΔΑ με κλειδί τον ΑΜΚΑ. Το πρόγραμμα θα περιλαμβάνει τις εξής λειτουργίες

- Εισαγωγή των στοιχείων του κάθε ατόμου στο αντίστοιχο ΔΔΑ ανάλογα με την ηλικία του (ΔΔΑ για άτομα με ηλικίες <=60 και ΔΔΑ για άτομα με ηλικίες > 60). Και στα 2 ΔΔΑ το κλειδί θα είναι ο ΑΜΚΑ.
- Εμφάνιση των 2 καταλόγων
- Αναζήτηση ατόμου με βάση τον ΑΜΚΑ και την ηλικία.

Δίνεται ένα στιγμιότυπο εκτέλεσης όπου φαίνεται πως θα γίνεται το διάβασμα των δεδομένων και η εμφάνιση του κάθε καταλόγου. Η αναζήτηση θα γίνει για 3 άτομα: α) ένα άτομο του οποίου τα στοιχεία έχουν καταχωρηθεί σε ένα εκ των 2 καταλόγων (ΑΜΚΑ, ηλικία), β) ένα άτομο του οποίου το ΑΜΚΑ του έχει καταχωρηθεί στον αντίστοιχο κατάλογο αλλά η ηλικία του δεν ταυτίζεται με τη δοθείσα και γ) ένα άτομο του οποίου το ΑΜΚΑ δεν έχει καταχωρηθεί σε κανένα κατάλογο. Τα μηνύματα της κάθε περίπτωσης φαίνονται στο στιγμιότυπο εκτέλεσης.

(για την απλοποίηση της εισαγωγής δεδομένων δόθηκαν ίδιες τιμές για ΑΜΚΑ, ΑΦΜ, ηλικία)

```
Give AMKA? 20
Give AFM? 20
Give  age? 20

Continue Y/N: y
```

Give AMKA? 15
Give AFM? 15
Give age? 15

Continue Y/N: y
Give AMKA? 30
Give AFM? 30
Give age? 30

Continue Y/N: y
Give AMKA? 65
Give AFM? 65
Give age? 65

Continue Y/N: y
Give AMKA? 62
Give AFM? 62
Give age? 62

Continue Y/N: y
Give AMKA? 70
Give AFM? 70
Give age? 70

Continue Y/N: n

People with age less-equal 60
(15, 15, 15) (20, 20, 20) (30, 30, 30)

People with age greater than 60
(62, 62, 62) (65, 65, 65) (70, 70, 70)
Give AMKA: 20
Give age: 20
The person with AMKA 20, AFM 20 and age 20 is in the catalogue
Give AMKA: 70
Give age: 71
The person with AMKA 70 has age 70 different from the person you are looking for
Give AMKA: 80
Give age: 90
The person with AMKA 80 and age 90 is Unknown

30. Το αρχείο transactions.txt περιλαμβάνει ύποπτες συναλλαγές και θέλουμε να εντοπίσουμε τις m μεγαλύτερες συναλλαγές. Η διαθέσιμη μνήμη δεν επαρκεί για να διαβάσουμε όλες τις συναλλαγές από το αρχείο και να τις αποθηκεύσουμε σε μια δομή δεδομένων στη μνήμη. Επιλέξτε την κατάλληλη δομή δεδομένων ώστε να βρίσκει τις m μεγαλύτερες συναλλαγές. Μετά την εύρεση των m μεγαλύτερων συναλλαγών θα εμφανίζει το μέγεθος και τα στοιχεία της δομής δεδομένων (προσαρμόστε κατάλληλα την PrintHeap) και στη συνέχεια θα εμφανίσει σε αύξουσα διάταξη τις m μεγαλύτερες συναλλαγές. Την τιμή της m θα τη δίνει ο χρήστης, και θεωρήστε ότι δίνεται τιμή πολύ μικρότερη του μεγέθους του αρχείου χωρίς να γίνεται σχετικός έλεγχος. Ως δομή δεδομένων θα πρέπει να χρησιμοποιηθεί σωρός (μέγιστος ή ελάχιστος σωρός;). Δίνεται ένα στιγμιότυπο εκτέλεσης:

Give m: 5
Data Structure size =5
4121.85 4409.74 4381.21 4747.08 4732.35
Transactions
4121.85 4381.21 4409.74 4732.35 4747.08

31. Αλλάξτε τη συνάρτηση αναζήτησης BSTSearch έτσι ώστε να υπολογίζει και να επιστρέφει και τον αριθμό των κόμβων που προσπέλασε κατά την αναζήτηση ενός στοιχείου:

void BSTSearch(BinTreePointer Root, BinTreeElementType KeyValue, boolean *Found, BinTreePointer *LocPtr, int *nodes)

Γράψτε κυρίως πρόγραμμα το οποίο αρχικά θα δημιουργεί ένα ΔΔΑ από το αρχείο i31f5.txt στο οποίο καταχωρούνται τα στοιχεία των φοιτητών ενός τμήματος, που αποτελούνται από:

Αριθμός Μητρώου (ΑΜ)	*Αλφαριθμητικό 8 χαρακτήρες*
Όνομα	*Αλφαριθμητικό 10 χαρακτήρες*
Επώνυμο	*Αλφαριθμητικό 10 χαρακτήρες*

Το ΔΔΑ θα δημιουργείται βάση του ΑΜ των φοιτητών. Η δημιουργία του ΔΔΑ από το αρχείο i31f5.txt θα γίνεται μέσω της συνάρτησης void BuildBST(BinTreePointer *Root);

Στη συνέχεια, το πρόγραμμα θα διαβάζει από τον χρήστη 3 ΑΜ και θα τα αναζητά με τη χρήση της τροποποιημένης BSTSearch στο ΔΔΑ, και θα εμφανίζει το πλήθος των κόμβων που προσπελάστηκαν κατά την αναζήτηση, καθώς και όλα τα στοιχεία του φοιτητή αν αυτός βρεθεί στο ΔΔΑ.

Δίνεται ένα στιγμιότυπο εκτέλεσης:

```
Give am:mac10005
komboi=3 AM=mac10005 Onoma=dimitra Epwnymo=pappa
Give am:mac10001
komboi=1 AM=mac10001 Onoma=nikos Epwnymo=georgiou
Give am:el11111
komboi=4 o foiththns den brethike sto DDA
```

32. Υλοποιήστε την παρακάτω συνάρτηση:

int LeftNodeCount(BinTreePointer Root);

η οποία δέχεται σαν είσοδο τη ρίζα ενός ΔΔΑ (Root) και επιστρέφει το πλήθος των αριστερών κόμβων-παιδιών του ΔΔΑ. Στο κυρίως πρόγραμμα θα δημιουργείτε δύο ΔΔΑ. Ένα με εισαγωγή των γραμμάτων της λέξης "ALGORITHM" με τη σειρά που εμφανίζονται στην λέξη, και ένα με τα γράμματα να εισάγονται με την ανάποδη σειρά (δλδ. M, H, T, I, ..κτλ). Στη συνέχεια, θα εμφανίζονται τα στοιχεία των δύο δέντρων χρησιμοποιώντας την ενδοδιατεταγμένη διάσχιση, και τέλος θα καλείται για κάθε δέντρο η συνάρτηση LeftNodeCount και θα εμφανίζεται το αποτέλεσμα της.

Ακολουθεί στιγμιότυπο εκτέλεσης.

```
First tree with inorder traverse
A G H I L M O R T
Second tree with inorder traverse
A G H I L M O R T
First tree LeftNodeCount = 3
Second tree LeftNodeCount = 5
```

33. Υλοποιήστε δύο συναρτήσεις *MinBSTValue* και *MaxBSTValue* που δέχονται ως μοναδική παράμετρο την ρίζα ενός ΔΔΑ και επιστρέφουν τη μικρότερη και τη μεγαλύτερη τιμή του αντίστοιχα. Αν το ΔΔΑ είναι κενό θα επιστρέφουν -1. Για να ελέγξετε την ορθότητα των συναρτήσεων, στο κυρίως πρόγραμμα:

- Θα διαβάζονται θετικοί ακέραιοι αριθμοί μέχρι να διαβαστεί το -1, και θα κατασκευάζονται δύο ΔΔΑ, ένα για τους περιττούς και ένα για τους άρτιους αριθμούς που διαβάστηκαν. Δεν χρειάζεται να γίνει έλεγχος για την εγκυρότητα των αριθμών.
- Θα καλούνται οι συναρτήσεις *MinBSTValue* και *MaxBSTValue* και θα εμφανίζονται οι τιμές που επιστρέφουν, πρώτα για το ΔΔΑ των περιττών αριθμών και στην συνέχεια των άρτιων.

Υπόδειξη: Σε καμία από τις δύο συναρτήσεις δεν χρειάζεται να διασχίσετε όλο το ΔΔΑ. Στην περίπτωση εύρεσης της ελάχιστης τιμής, για παράδειγμα, λόγω των ιδιοτήτων ενός ΔΔΑ μπορείτε απλά να ακολουθήσετε το μονοπάτι που καθορίζουν οι αριστεροί δείκτες μέχρι κάποιος από αυτούς να δείχνει σε κενό υποδέντρο.

Ακολουθεί στιγμιότυπο εκτέλεσης:

```
Dose enan thetiko akeraio:5
Dose enan thetiko akeraio:2
Dose enan thetiko akeraio:9
Dose enan thetiko akeraio:11
Dose enan thetiko akeraio:3
Dose enan thetiko akeraio:22
Dose enan thetiko akeraio:45
Dose enan thetiko akeraio:48
Dose enan thetiko akeraio:64
Dose enan thetiko akeraio:1
Dose enan thetiko akeraio:8
Dose enan thetiko akeraio:4
Dose enan thetiko akeraio:16
Dose enan thetiko akeraio:-1
H mikroteri timi toy DDA twn perittvn arithmwn einai: 1
H megalyteri timi toy DDA twn perittvn arithmwn einai: 45
```

H mikroteri timi toy DDA twn artiwn arithmwn einai: 2
H megalyteri timi toy DDA twn artiwn arithmwn einai: 64

34. Υλοποιήστε **μη αναδρομική** συνάρτηση NonRecTraversal η οποία δέχεται σαν είσοδο τη ρίζα ενός ΔΔΑ και θα διασχίζει και θα εμφανίζει όλα τα στοιχεία ενός ΔΔΑ. Συγκεκριμένα, ξεκινώντας από την ρίζα, θα εμφανίζει πρώτα την ρίζα, και μετά με τη σειρά το αριστερό και το δεξί της υποδέντρο (προδιατεταγμένη διάσχιση).
Για να ελέγξετε την ορθότητα της συνάρτησης σας στο κυρίως πρόγραμμα θα δημιουργείτε δύο ΔΔΑ. Ένα με εισαγωγή των γραμμάτων της λέξης "ALGORITHM" με τη σειρά που εμφανίζονται στην λέξη, και ένα με τα γράμματα να εισάγονται με την ανάποδη σειρά (δλδ. M, H, T, I, ..κτλ). Στη συνέχεια, θα καλείται η συνάρτηση για τα δύο δέντρα με τη σειρά.
Υπόδειξη: Χρησιμοποιήστε βοηθητική στοίβα στην οποία θα εισάγεται κάθε φορά πρώτα την ρίζα, και μετά τα παιδιά της.
Ακολουθεί στιγμιότυπο εκτέλεσης:
First tree
A L G I H O M R T
Second tree
M H G A I L T R O

35. Υλοποιήστε την παρακάτω συνάρτηση: BinTreePointer LCANode(BinTreePointer Root, BinTreePointer node1, BinTreePointer node2); η οποία δέχεται σαν είσοδο τη ρίζα ενός ΔΔΑ (Root) και δύο κόμβους, και επιστρέφει τον χαμηλότερο κοινό πρόγονο (Lowest Common Ancestor, LCA) των δύο κόμβων στο ΔΔΑ, δηλαδή τον κόμβο στο πιο χαμηλό επίπεδο του δέντρου που έχει και τους δύο κόμβους ως απογόνους. Στο κυρίως πρόγραμμα, θα δημιουργείται ένα ΔΔΑ με ακέραιους αριθμός. Συγκεκριμένα, θα διαβάζεται πρώτα το πλήθος των στοιχείων προς εισαγωγή στο ΔΔΑ, και μετά 1-1 τα στοιχεία αυτά. Στη συνέχεια, θα εμφανίζονται τα στοιχεία του δέντρου με ενδοδιατεταγμένη διάσχιση, και μετά θα διαβάζονται τα δύο στοιχεία για τα οποία θα πρέπει να βρεθεί ο LCA. Αν οποιοδήποτε από τα δύο στοιχεία δεν υπάρχουν στο ΔΔΑ, θα πρέπει και τα δύο να διαβάζονται ξανά. Όταν διαβαστούν έγκυρα στοιχεία, θα καλείται η συνάρτηση LCANode και θα εμφανίζεται το αποτέλεσμα της. Ακολουθεί στιγμιότυπο εκτέλεσης.

Enter number of elements for the tree: 8
Enter number: 20
Enter number: 15
Enter number: 42
Enter number: 39
Enter number: 14
Enter number: 56
Enter number: 72
Enter number: 19
Elements of BST in increasing order
14 15 19 20 39 42 56 72
Give first element of BST:72
Give second element of BST:39
LCA node of 72 and 39 is node with element 42

36. Υλοποιήστε την παρακάτω συνάρτηση: boolean Search(HeapType Heap, HeapElementType ArgKey, int *left, int *right); η οποία δέχεται σαν είσοδο ένα σωρό και ένα στοιχείο ArgKey, αναζητά το στοιχείο στον σωρό και επιστρέφει TRUE ή FALSE αντίστοιχα. Αν το στοιχείο βρεθεί η Search επιστρέφει επίσης τις θέσεις στον πίνακα του σωρού που είναι αποθηκευμένα τα παιδιά του κόμβου με κλειδί την τιμή του στοιχείου. Αν δεν υπάρχει κόμβος-παιδί, επιστρέφεται -1 για τη θέση του. Αν το στοιχείο ArgKey υπάρχει πάνω από μία φορά στον σωρό, επιστρέφεται ο κόμβος που αντιστοιχεί στην πρώτη του εμφάνιση. Στο κυρίως πρόγραμμα, θα δημιουργείται ένας σωρός με 8 στοιχεία, χαρακτήρες, και θα εμφανίζεται ο σωρός που δημιουργήθηκε. Στη συνέχεια θα καλείται για 4 φορές η συνάρτηση Search και θα εμφανίζονται τα παιδιά (και οι θέσεις τους) του στοιχείου που αναζητήθηκε αν αυτό βρέθηκε, και αντίστοιχο μήνυμα διαφορετικά. Ακολουθεί στιγμιότυπο εκτέλεσης.

Enter a letter : H
Enter a letter : K
Enter a letter : A
Enter a letter : N
Enter a letter : O
Enter a letter : H
Enter a letter : D
Enter a letter : R
Size=8
N L R
R, O, H

O, N, K
H, A, D
N, H,
Heap array
R O H N K A D H
Enter a letter : H
L=6, R=7
A D
Enter a letter : K
L=-1, R=-1

Enter a letter : N
L=8, R=-1
H
Enter a letter : F
Not found

.