ΕΛΛΗΝΙΚΟ ΜΕΣΟΓΕΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ
ΕΡΓΑΣΤΗΡΙΟ

ΗΡΑΚΛΕΙΟ 2020

ΕΙΣΑΓΩΓΗ

(ΓΕΝΙΚΑ ΓΙΑ ΤΟ ΕΡΓΑΣΤΗΡΙΟ – ΜΑΘΗΜΑΤΑ – ΕΞΕΤΑΣΗ)

Το φυλλάδιο αυτό αφορά το εργαστήριο του μαθήματος «Δομές Δεδομένων» και περιλαμβάνει ασκήσεις, οι οποίες πρέπει να εκτελεστούν στη διάρκεια του εργαστηρίου, υποδείξεις επί των ασκήσεων, καθώς και προτεινόμενες ασκήσεις για περαιτέρω άσκηση.

Συνιστάται να προετοιμάζετε τις ασκήσεις πριν το εργαστήριο, στην διάρκειά του δε να ασχολείστε με την επίλυση συγκεκριμένων δυσκολιών και αποριών.

Είναι επίσης απαραίτητο να διαβάζετε τις «Επεξηγήσεις – υπενθυμίσεις» οι οποίες σας δίνονται πριν κάθε άσκηση. Αυτές αποτελούν τις ελάχιστες γνώσεις θεωρίας που πρέπει να έχετε πριν προχωρήσετε στην υλοποίηση των προγραμμάτων που σας ζητούνται.

<u>Προσοχή!</u> Στα εργαστήρια δεν επαναλαμβάνεται η θεωρία! Γίνεται μόνο υπενθύμιση ορισμένων σημείων. Αυτό σημαίνει ότι η προετοιμασία και η παρακολούθηση του μαθήματος της θεωρίας είναι απαραίτητη. Σημεία της θεωρίας θα θίγονται μόνο όταν υπάρχει κάποια χρονική «ασυμφωνία» στην διδασκόμενη ύλη, όταν δηλαδή κάποιο θέμα δεν έχει ήδη καλυφθεί στο μάθημα.

Στη διάρκεια του εξαμήνου γίνονται (τουλάχιστον) 6 εργαστήρια.

Στη διάρκεια των εργαστηρίων διεξάγονται διαγωνίσματα (τεστ). Τον αριθμό, τη διάρκεια και την βαρύτητα καθενός τέστ για τη διαμόρφωση του τελικού βαθμού του εργαστηρίου, τα καθορίζει ο διδάσκων.

ΕΡΓΑΣΤΗΡΙΟ 1

Α΄. ΠΕΡΙΕΧΟΜΕΝΑ ΜΑΘΗΜΑΤΟΣ

Στο εργαστήριο 1 καλύπτονται τα παρακάτω θέματα:

- Πίνακες μιας διάστασης.
- Μονοδιάστατοι πίνακες ως ορίσματα συναρτήσεων.
- Πίνακες δύο διαστάσεων.
- Πίνακες συμβολοσειρών.
- Συμμετρικοί, τριγωνικοί πίνακες.
- Αραιοί πίνακες.

Β’. ΑΣΚΗΣΕΙΣ ΓΙΑ ΕΚΤΕΛΕΣΗ – ΕΠΙΔΕΙΞΗ / ΕΠΕΞΗΓΗΣΕΙΣ

Να εκτελεστούν τουλάχιστον οι ασκήσεις 4, 8, 11.

Μονοδιάστατοι πίνακες γενικά.

Επεξηγήσεις – υπενθυμίσεις (ασκήσεις 1, 2) :

- Θυμηθείτε ότι για ένα πίνακα pin ισχύουν τα εξής:

  pin == &pin[0]

  pin+k == &pin[k]

  ***pin == pin[0]**

  ***(pin+k) == pin[k]**

- Η **τιμή ενός δείκτη** ισούται με τη διεύθυνση μνήμης του byte στο οποίο είναι τοποθετημένος ο δείκτης και εμφανίζεται στην οθόνη με την χρήση του **προσδιοριστή %p**.

1. Χρησιμοποιώντας συμβολισμό δεικτών να δώσετε τιμές από το πληκτρολόγιο σε ένα πίνακα ακεραίων Ν θέσεων. Στη συνέχεια να εμφανίσετε στην οθόνη τις διευθύνσεις και τις τιμές των στοιχείων του πίνακα.

2. Να γραφεί ένα πρόγραμμα, το οποίο να διαβάζει ένα ακέραιο και να τον μετατρέπει στον αντίστοιχο δυαδικό.

Η μετατροπή γίνεται με τη μέθοδο των συνεχών διαιρέσεων. Ο αριθμός δηλαδή διαιρείται συνεχώς με το 2. Όσο το πηλίκο είναι διάφορο του μηδενός παίρνουμε το υπόλοιπο αυτής της ακέραιας διαίρεσης. Τα υπόλοιπα να καταχωρούνται σε ένα πίνακα ακεραίων, Ν θέσεων. Ο δυαδικός αριθμός αποτελείται από αυτά τα υπόλοιπα, αλλά από το τέλος προς την αρχή. Να ορίσετε το Ν αρκετά μεγάλο, ώστε ο πίνακας να «χωρέσει» τον μέγιστο δυαδικό αριθμό που θα προκύψει.

Επεξηγήσεις – υπενθυμίσεις (ασκήσεις 3, 4, 5, 6, 7) :

- Για να «περάσω» **σε μια συνάρτηση ως παράμετρο ένα πίνακα**, **περνάω ένα δείκτη στην αρχή του πίνακα και** (αν χρειάζεται) **το μέγεθος του πίνακα**.
- Στον ορισμό μιας συνάρτησης (της parad) **οι παρακάτω συμβολισμοί είναι ισοδύναμοι**:

  void parad (int *pin)

  void parad (int pin[ ])

  Σε κάθε περίπτωση, **το pin είναι δείκτης σε ακέραιο.**

Μονοδιάστατοι πίνακες ως ορίσματα συνάρτησης.

3. Μελετείστε το παρακάτω πρόγραμμα, στο οποίο χρησιμοποιούμε τρείς συναρτήσεις. Μέσω της fstore( ) τοποθετούμε δεδομένα στον πίνακα array. Μέσω της fretrieve( ) τα τυπώνουμε στην οθόνη και μέσω της fedit( ) μας δίνεται η δυνατότητα να αλλάξουμε τα στοιχεία που είχαμε εισαγάγει με την fstore( ).

```
#define N 5

void fstore (int [ ]);
void fretrieve (int [ ]);
void fedit (int [ ]);

main( ) {
    int array[N];

    printf("Balte times ston pinaka:\n");

    fstore(array);

    fretrieve(array);
```

```
    fedit(array);
    fretrieve(array);
    return 0; }


void fstore (int a[ ]) {
    int i;

    for (i=0; i<N; i++)
        scanf("%d", &a[i]); }

void fretrieve (int a[ ]) {
    int i;

    for (i=0; i<n; i++)
        printf("%6d", a[i]);
    printf("\n"); }

 void fedit (int a[ ]) {
     int i, q;

     for (i=0; i<n; i++) {
         printf ("Prev.data:%d\nEnter 1 to edit 0 to skip", a[i]);
         scanf ("%d", &q);
         if(q == 1) {
             printf("Enter new Value: ");
             scanf("%d", &a[i]); } } }
```

4. Να γραφεί πρόγραμμα, στο οποίο διαβάζονται ακέραιοι από το πληκτρολόγιο και καταχωρούνται στον πίνακα list. Το διάβασμα συνεχίζεται μέχρι να γεμίσει ο πίνακας ή μέχρι να δοθεί ακέραιος ίσος με μηδέν. Όταν τελειώσει το διάβασμα, στον πίνακα θα έχουν τοποθετηθεί συνολικά k ακέραιοι διάφοροι του μηδενός. Στη συνέχεια καλείται μια συνάρτηση, η max( ), η οποία βρίσκει το μέγιστο μη μηδενικό στοιχείο (meg) του πίνακα αυτού και επιστρέφει το μέγιστο αυτό στοιχείο στη main( ).

5. Να γραφεί πρόγραμμα, στο οποίο να αθροίζονται μεταξύ τους τα στοιχεία δύο πινάκων. Να δημιουργηθούν τρείς συναρτήσεις. Η πρώτη input( ) χρησιμοποιείται για να βάλουμε τιμές στους δύο πίνακες, η δεύτερη addarray( ) για να γίνει η πρόσθεση και η τρίτη display( ) για να εμφανίσουμε στην οθόνη τον τρίτο πίνακα με το αποτέλεσμα της πρόσθεσης.

6. Να γραφεί πρόγραμμα, στο οποίο να διαβάζονται float από το πληκτρολόγιο και καταχωρούνται σε ένα πίνακα, τον c. Οι float αυτοί υποτίθεται ότι αντιπροσωπεύουν ηλικίες κάποιων ατόμων. Στη συνέχεια, θα καλείται μια συνάρτηση, η avg( ), η οποία θα βρίσκει τον μέσο όρο των ηλικιών και θα τον επιστρέφει στην main( ), όπου και θα εμφανίζεται στην οθόνη.

7.

Επεξηγήσεις – υπενθυμίσεις (άσκηση 7) :

Υπάρχουν διάφοροι αλγόριθμοι ταξινόμησης ενός πίνακα. Αυτός που περιγράφεται εδώ είναι γνωστός ως **«Ταξινόμηση με επιλογή»**.

Η συνάρτηση θα ξεκινά από το στοιχείο της πρώτης θέσης του πίνακα, το list[0], με στόχο να τοποθετηθεί στη θέση αυτή η μικρότερη τιμή του πίνακα. Η συνάρτηση να διατρέχει όλα τα υπόλοιπα στοιχεία, από το list[1] μέχρι το list[N-1] και να συγκρίνει καθένα με το πρώτο. Αν βρει κάποιο μικρότερο από το πρώτο, τα στοιχεία εναλλάσσονται μεταξύ τους. Τελικά το list[0] θα έχει τη μικρότερη τιμή του πίνακα.

Αφού τελειώσουμε με το πρώτο στοιχείο επαναλαμβάνεται η διαδικασία, προσπαθώντας να βάλουμε στη θέση 1 του πίνακα τη δεύτερη σε μέγεθος τιμή. Συγκρίνονται δηλαδή όλα τα στοιχεία από το list[2] και μετά με το list[1]. Αν βρεθεί κάποιο στοιχείο μικρότερο από το list[1], τα στοιχεία εναλλάσσονται μεταξύ τους κ.ο.κ.=

Χρειάζεστε δύο εμφωλευμένες επαναλήψεις.

Έστω ένας πίνακας ακεραίων, ο list, με N θέσεις. Ο πίνακας να πάρει τιμές στη main( ) από το πληκτρολόγιο. Να καλείται μια συνάρτηση, η sort( ), η οποία θα ταξινομήσει τα στοιχεία του πίνακα. Ο ταξινομημένος πίνακας να εμφανίζεται στην οθόνη από την main( ).

Δισδιάστατοι πίνακες ως ορίσματα συνάρτησης.

8. Να γραφεί ένα πρόγραμμα, στο οποίο να δηλώσετε ένα δισδιάστατο πίνακα ακεραίων, τον **grades[ ][ ]**. Ο πίνακας να αποτελείται το πολύ από τόσες γραμμές, όσοι είναι οι μαθητές μιας τάξης και από 5 στήλες, όσος είναι ο αριθμός των τεστ, στα οποία έχουν υποβληθεί οι μαθητές

Να δώσετε τιμές στον πίνακα grades και στη συνέχεια να δημιουργήσετε μια συνάρτηση, την **better( )**, η οποία να βρίσκει και να επιστρέφει στη main( ) τον

μεγαλύτερο από τους βαθμούς αυτούς, ο οποίος και να εμφανίζεται στην οθόνη. Ο πίνακας grades δεν είναι απαραίτητο να γεμίσει ολόκληρος ως προς τον αριθμό των μαθητών. Όταν καλείται η συνάρτηση better( ), αυτή να ενημερώνεται για το πόσες γραμμές του πίνακα να ερευνήσει (δηλαδή για το πόσοι μαθητές εξετάστηκαν στα τεστ).

9.

Επεξηγήσεις – υπενθυμίσεις (άσκηση 9) :

Με δεδομένη την δήλωση: **char students[R][C];**

- Η **k γραμμή** του πίνακα είναι μονοδιάστατος πίνακας χαρακτήρων και λέγεται **students[k]**.
- Το **students[k]** είναι πίνακας χαρακτήρων, άρα και **δείκτης σε χαρακτήρα**.
- Η **ταξινόμηση** (κατάταξη αλφαβητικά) **συμβολοσειρών** μπορεί να γίνει με τον ίδιο αλγόριθμο που γίνεται η ταξινόμηση ακεραίων. Η **σύγκριση των συμβολοσειρών** θα γίνεται με την χρήση της συνάρτησης **strcmp( )**.

Στη main( ) ενός προγράμματος να δηλώσετε ένα δισδιάστατο πίνακα χαρακτήρων, RxC, τον **students**. Στον πίνακα αυτόν να καταχωρήσετε R ονόματα φοιτητών (συμβολοσειρές) από το πληκτρολόγιο. Στη συνέχεια:

- Να εμφανίσετε στην οθόνη τα ονόματα με τη σειρά που τα καταχωρήσατε στον πίνακα.
- Να κατατάξετε τα ονόματα στον πίνακα αλφαβητικά και να τα εμφανίσετε ξανά στην οθόνη.

10.

Επεξηγήσεις – υπενθυμίσεις (άσκηση 10) :

- Ο δισδιάστατος πίνακας pin λέγεται **συμμετρικός**, εάν:
  i) Είναι **τετραγωνικός**, δηλ. έχει ίσο αριθμό γραμμών και στηλών.
  ii) Το στοιχείο της **j γραμμής και της k στήλης** είναι **ίσο** με το στοιχείο της της **k γραμμής και της j στήλης**. Αυτό ισχύει **για κάθε k και j**.

Επεξηγήσεις – υπενθυμίσεις (άσκηση 10, συνέχεια) :

- Ο δισδιάστατος πίνακας pin λέγεται **κάτω τριγωνικός**, εάν:
  i) Είναι **τετραγωνικός**, δηλ. έχει ίσο αριθμό γραμμών και στηλών.
  ii) Κάθε **στοιχείο πάνω από την κύρια διαγώνιο είναι ίσο με μηδέν** (δηλαδή τα στοιχεία με ετικέτες (i, j), όπου j>i).
- Ο δισδιάστατος πίνακας pin λέγεται **άνω τριγωνικός**, εάν:
  i) Είναι **τετραγωνικός**, δηλ. έχει ίσο αριθμό γραμμών και στηλών.
  ii) Κάθε **στοιχείο κάτω από την κύρια διαγώνιο είναι ίσο με μηδέν** (δηλαδή τα στοιχεία με ετικέτες (i, j), όπου j<i).
- Ένας πίνακας pin λέγεται **αραιός**, εάν **τουλάχιστον το 80% των στοιχείων του είναι ίσα με μηδέν**.

Στη main( ) ενός προγράμματος να δηλώσετε ένα δισδιάστατο πίνακα ακεραίων NxN, τον **pinax**. Να γεμίσετε τον πίνακα από το πληκτρολόγιο. Να ορίσετε και να καλέσετε τρείς συναρτήσεις, οι οποίες να διαπιστώνουν εάν ο **pinax** είναι:

(α) Συμμετρικός
(β) Άνω τριγωνικός ή κάτω τριγωνικός
(γ) Αραιός

Το αποτέλεσμα της συνάρτησης να γράφεται από τη main( ) στην οθόνη.

11.

Επεξηγήσεις – υπενθυμίσεις (άσκηση 11) :

- Με δεδομένη την δήλωση: **char pin[R][C];**
  - Η **k γραμμή** του πίνακα είναι μονοδιάστατος πίνακας χαρακτήρων και λέγεται **pin[k]**.
  - Το **pin[k]** είναι πίνακας χαρακτήρων, άρα και **δείκτης σε χαρακτήρα**.
- **Κενή συμβολοσειρά** είναι εκείνη στην οποία ο πρώτος χαρακτήρας είναι το \x0 ή αλλιώς, το μήκος της είναι ίσο με μηδέν. Τέτοια συμβολοσειρά προκύπτει όταν, περιμένοντας να διαβάσουμε συμβολοσειρά από από το πληκτρολόγιο, δώσουμε <enter>.

Στη main( ) ενός προγράμματος να δηλώσετε ένα δισδιάστατο πίνακα χαρακτήρων, RxC, τον pin. Στον πίνακα αυτόν να καταχωρείτε συμβολοσειρές.

Θα καταχωρήσετε R το πολύ συμβολοσειρές ή αλλιώς θα σταματάτε όταν δώσετε κενή συμβολοσειρά. Να εμφανίσετε στην οθόνη το σύνολο των χαρακτήρων που δόθηκαν, καθώς και το πλήθος των χαρακτήρων της μεγαλύτερης συμβολοσειράς.

12.

Επεξηγήσεις – υπενθυμίσεις (άσκηση 12) :

Ο δισδιάστατος πίνακας pinax λέγεται **ταυτοτικός**, εάν:

i) Είναι **τετραγωνικός**, δηλ. έχει ίσο αριθμό γραμμών και στηλών.

ii) **Κάθε στοιχείο της κύριας διαγωνίου** του είναι **ίσο με 1,** ενώ **όλα τα υπόλοιπα στοιχεία είναι ίσα με μηδέν** (δηλαδή τα στοιχεία με ετικέτες (i, j), όπου j==i είναι 1).

Στη main( ) ενός προγράμματος να δηλώσετε ένα δισδιάστατο πίνακα ακεραίων NxN, τον **pinax**. Να γεμίσετε τον πίνακα από το πληκτρολόγιο. Να ορίσετε και να καλέσετε μια συνάρτηση, η οποία να διαπιστώνει εάν ο **pinax** είναι ταυτοτικός. (Ταυτοτικός είναι ο πίνακας, στον οποίο τα στοιχεία της κύριας διαγωνίου είναι 1, ενώ όλα τα υπόλοιπα είναι μηδέν).

13.

Επεξηγήσεις – υπενθυμίσεις (άσκηση 6) :

Ο δισδιάστατος πίνακας pinax λέγεται **αντισυμμετρικός**, εάν:

i) Είναι **τετραγωνικός**, δηλ. έχει ίσο αριθμό γραμμών και στηλών.

ii) **Κάθε στοιχείο της κύριας διαγωνίου** του είναι **ίσο με μηδέν,** ενώ **για τα τα υπόλοιπα στοιχεία του** ισχύει**:**

pinax[j][k] == - pinax[k][j]

Στη main( ) ενός προγράμματος να δηλώσετε ένα δισδιάστατο πίνακα ακεραίων NxN, τον **pinax**. Να γεμίσετε τον πίνακα από το πληκτρολόγιο. Να ορίσετε και να καλέσετε μια συνάρτηση, η οποία να διαπιστώνει εάν ο **pinax** είναι αντισυμμετρικός.

ΕΡΓΑΣΤΗΡΙΟ 2

Α΄. ΠΕΡΙΕΧΟΜΕΝΑ ΜΑΘΗΜΑΤΟΣ

Στο εργαστήριο 2 καλύπτονται τα παρακάτω θέματα:

- Δομές (structures): Περιγραφή, πεδία δομής, δηλώσεις και δεδομένα
- Φωλιασμένες δομές.
- Πίνακες δομών.
- Δομές ως παράμετροι και ως τιμή επιστροφής συναρτήσεων.
- Δείκτες σε δομές.
- Συναρτήσεις malloc, calloc, realloc.
- Πίνακες δεικτών.
- Δυναμικοί πίνακες,

Β΄. ΑΣΚΗΣΕΙΣ ΓΙΑ ΕΚΤΕΛΕΣΗ – ΕΠΙΔΕΙΞΗ / ΕΠΕΞΗΓΗΣΕΙΣ

Να εκτελεστούν τουλάχιστον οι ασκήσεις 5, 11, 14.

Δομές: Περιγραφή και πεδία. Δηλώσεις και δεδομένα.

Επεξηγήσεις – υπενθυμίσεις (γιά όλες τις ασκήσεις) :

- Η **περιγραφή της δομής** προϋπάρχει της δήλωσης για να «διδάξει» στον compiler πώς είναι το νέο είδος μεταβλητών που δημιουργούμε.
- Η **περιγραφή της δομής** βρίσκεται **μέσα σε άγκιστρα**, **μετά τα οποία** υπάρχει το **;**
- Στο κάθε πεδίο μιας δομής αναφερόμαστε ως εξής:

  Όνομα δομής . Όνομα πεδίου
- Η εμφάνιση **μόνο του είδους** της δομής (της λέξης δηλαδή που ακολουθεί την λέξη struct) αποτελεί **συντακτικό λάθος**.
- Η εμφάνιση **μόνο του ονόματος κάποιου πεδίου της δομής** (των λέξεων δηλαδή που εμφανίζονται στην περιγραφή μιας δομής) αποτελεί επίσης **συντακτικό λάθος.**
- Μια δομή a κάποιου τύπου μπορεί **να γίνει ίση** με μια δομή b **του ίδιου τύπου** με **απλή απόδοση τιμής**, δηλαδή:

  a = b;

1. Σε ένα πρόγραμμα έχετε τον εξής τύπο δομών:

```
struct stype {
    Int  j;
    char ch[30];
    float fp; };
```

Να δηλώσετε στη main( ) δυο μεταβλητές δομές του πιο πάνω τύπου, τις str1 και str2 και να τους δώσετε τιμές από το πληκτρολόγιο. Στα πεδία ch να καταχωρήσετε συμβολοσειρές. Στη συνέχεια να εναλλάξετε τα περιεχόμενα των str1 και str2 και να γράψετε τα πεδία τους στην οθόνη.

Φωλιασμένες δομές.

Επεξηγήσεις – υπενθυμίσεις (άσκηση 2) :

- **Φωλιασμένες** λέγονται οι δομές των οποίων **κάποιο ή κάποια πεδία** είναι **δομή**, η οποία **έχει ήδη περιγραφεί** προηγουμένως.
- Η **προσπέλαση των πεδίων στις φωλιασμένες δομές** γίνεται με την **πολλαπλή χρήση της τελείας (.)**.

2. Να γράψετε ένα πρόγραμμα, στο οποίο να περιγράψετε δύο είδη δομών, τις one και two. Οι δομές του είδους one έχουν δύο πεδία: το ak (ακέραιος) και το pin (πίνακας χαρακτήρων 30 θέσεων). Οι δομές του είδους two έχουν τέσσερα πεδία: το data (ακέραιος), το mat (πίνακας ακεραίων 20 θέσεων), το item (δομή του είδους one) και το melos (δομή του είδους one). Να δηλώσετε δυο μεταβλητές, την person (δομή του είδους one) και την memb (δομή του είδους two). Να γράψετε τις εντολές (κάθε μια είναι ανεξάρτητη από τις υπόλοιπες), με τις οποίες θα γίνονται τα παρακάτω, μετά δε από κάθε εντολή να γράφονται στην οθόνη τα περιεχόμενα των δομών για επιβεβαίωση:

- Διαβάζουμε τιμές από το πληκτρολόγιο για τα πεδία των δομών person και memb. Στους πίνακες χαρακτήρων να τοποθετήσετε συμβολοσειρές.
- Αντιγράφουμε τη συμβολοσειρά που υπάρχει στον πίνακα pin της person στον αντίστοιχο πίνακα του πεδίου item της memb.
- Γράφουμε στην οθόνη το μήκος της συμβολοσειράς που υπάρχει στον πίνακα χαρακτήρων του πεδίου melos της δομής memb.
- Στο τέλος της συμβολοσειράς που υπάρχει στον πίνακα pin της person κάνουμε επικόλληση της συμβολοσειράς που υπάρχει στον αντίστοιχο

πίνακα χαρακτήρων του πεδίου melos, της δομής της memb, εφ' όσον υπάρχει αρκετός χώρος.

- Αντιγράφουμε το πεδίο item της δομής memb, στο πεδίο melos της ίδιας δομής.

Πίνακες δομών.

Επεξηγήσεις – υπενθυμίσεις (άσκηση 3) :

- Σε ένα πίνακα δομών, **κάθε στοιχείο του πίνακα είναι μια δομή**. Στην άσκηση 3, το pin είναι ένας πίνακας N δομών του είδους student. Άρα, το pin[0], το pin[1] κλπ είναι δομές του είδους student.
- **Ο χαρακτήρας** για παράδειγμα **της 3ης θέσης του πίνακα name του τέταρτου στοιχείου του πίνακα pin** λέγεται **pin[3] . name[2]**
- Το **fflush(stdin) αδειάζει τον buffer εισόδου** κι έτσι η gets δεν επηρρεάζεται από το <enter> της scanf που έχει προηγηθεί.

3. Πληκτρολογήστε και εκτελέστε το πιο κάτω πρόγραμμα και παρατηρείστε τι κάνει:

```
#include<stdio.h>
#define N 10
struct student { int am; char name[30]; };

main( ) {
    struct  student  pin[N];                    /* Πίνακας δομών */
    int k;

    for (k=0; k<N; k++) {                              /* Διάβασμα */
        printf ("Dwste AM tou spoudasth %d -> ", k+1);
        scanf ("%d", &pin[k].am );              /* Προσέξτε το & */
        fflush (stdin);
        printf ("Dwste onoma -> ");
        gets (pin[k].name); }
    for (k=0; k<N; k++) {                              /* Εκτύπωση */
        printf ("Spoudasths %3d\n", k+1);
        printf ("AM=%4d\n", pin[k].am);
        printf ("Arxiko gramma onomatos=%3c\n",
                    pin[k].name[0]);
        printf ("Onoma=%s \n", pin[k].name ); } }
```

Δομές ως παράμετροι και ως τιμή επιστροφής συναρτήσεων.

Επεξηγήσεις – υπενθυμίσεις (άσκηση 4) :

- Μια **συνάρτηση** μπορεί να **δέχεται ως ορίσματα δομές**, όπως και οποιονδήποτε άλλο τύπο δεδομένων. Π.χ.: Η επικεφαλλίδα μιας συνάρτησης που είναι void και δέχεται ως παραμέτρους δύο δομές, τις s1 και s2 του τύπου stype της άσκησης 4, θα είναι:

  void example (struct stype s1, struct stype s2)

- Η **κλήση** της συνάρτησης θα είναι:

  example (s1, s2);

4. Σε ένα πρόγραμμα έχετε τον εξής τύπο δομών:

```
struct stype {
    int  j;
    char ch[30];
    float fp; };
```

Να δηλώσετε στη main( ) δυο μεταβλητές δομές του πιο πάνω τύπου, τις str1 και str2 και να τους δώσετε τιμές από το πληκτρολόγιο. Στα πεδία ch να καταχωρήσετε συμβολοσειρές. Στη συνέχεια να καλέσετε μια συνάρτηση, η οποία θα λέγεται struct_swap( ). Η συνάρτηση θα καλείται μια μόνο φορά στη main( ) και θα εναλλάσσει τα περιεχόμενα των str1 και str2. Η συνάρτηση να δηλωθεί, να οριστεί.

5.

Επεξηγήσεις – υπενθυμίσεις (άσκηση 5) :

- Μια **συνάρτηση** μπορεί να **δέχεται ως όρισμα πίνακα δομών**, όπως και οποιονδήποτε άλλο πίνακα. Π.χ.: Η επικεφαλλίδα μιας συνάρτησης που είναι void και δέχεται ως παράμετρο ένα πίνακα δομών του τύπου funds της άσκησης 5, τον pin, θα είναι:

  void paradeigma (struct funds *pin) ή

  void paradeigma (struct funds pin[ ])

- Η **κλήση** της συνάρτησης θα είναι:

  paradeigma (pin);

Σε ένα πρόγραμμα έχετε τον εξής τύπο δομών:

```
struct funds {
    char name[20];
    float poson; };
```

Να δηλώσετε στη main( ) ένα πίνακα Ν δομών του πιο πάνω τύπου, τον persons και να του δώσετε τιμές από το πληκτρολόγιο. Στο πεδίο name να καταχωρήσετε συμβολοσειρές. Στη συνέχεια:

- Να διαβάσετε ένα χαρακτήρα στη main( ), τον ch.
- Να καλέσετε μια συνάρτηση, την athroisma( ), η οποία θα καλείται μια μόνο φορά στη main( ) και θα κάνει τα εξής:
  - Θα ελέγχει ποιες από τις συμβολοσειρές των στοιχείων του πίνακα δομών έχουν ως πρώτο γράμμα το ch.
  - Θα δημιουργεί το άθροισμα των πεδίων poson αυτών των στοιχείων του πίνακα.
  - Θα επιστρέφει το άθροισμα στη main( ).

Η main( ) να εμφανίζει στην οθόνη την τιμή που επέστρεψε η athroisma( ).

6.

Επεξηγήσεις – υπενθυμίσεις (άσκηση 6) :

- Μια **συνάρτηση** μπορεί να έχει **τιμή επιστροφής δομή**, όπως και οποιοδήποτε άλλο είδος δεδομένων. Π.χ.: Η επικεφαλλίδα μιας συνάρτησης που είναι void και δέχεται ως μια δομή του τύπου funds της άσκησης 6, την str, θα είναι:

  struct funds reading (struct funds str)

- Η **κλήση** της συνάρτησης θα είναι:

  x = reading (str);

  όπου το x είναι δομή της main( ).

Σε ένα πρόγραμμα έχετε τον εξής τύπο δομών:

```
struct funds {
    char name[20];
    float poson; };
```

Να δηλώσετε στη main( ) ένα πίνακα Ν δομών του πιο πάνω τύπου, τον persons. Για κάθε στοιχείο του πίνακα (άρα Ν φορές), να καλείτε μια συνάρτηση, έστω την reading( ), η οποία θα διαβάζει μια δομή του είδους funds, θα την επιστρέφει στην main( ), από όπου και θα καταχωρείται στον πίνακα.

Αφού γεμίσει ο πίνακας persons, να καλείται μια άλλη συνάρτηση, η display( ), η οποία θα γράφει τα περιεχόμενά του persons στην οθόνη.
Να μη χρησιμοποιείτε εξωτερικές μεταβλητές.

Δείκτες σε δομές.

7.

Επεξηγήσεις – υπενθυμίσεις (άσκηση 7) :

- Στα **πεδία μιας δομής** στην οποία **δείχνει ένας δείκτης** αναφερόμαστε ως εξής:

  Όνομα δείκτη -> Όνομα πεδίου

- Σε μια παράσταση όπου υπάρχουν τελεστές **.** και **-> όλη η παράσταση αριστερά από την . πρέπει να είναι δομή**, ενώ **όλη η παράσταση αριστερά από το -> πρέπει να είναι δείκτης σε δομή**. Σε κάθε άλλη περίπτωση υπάρχει συντακτικό λάθος
- **Επικόλληση** μιας συμβολοσειράς σε άλλη γίνεται με την χρήση της συνάρτησης **strcat( )**.
  - Η strcat( ) δέχεται ως **ορίσματα δύο δείκτες σε χαρακτήρα**. **Επικολλά την συμβολοσειρά που ξεκινάει από εκεί που δείχνει ο δεύτερος δείκτης, στο τέλος της συμβολοσειράς που δείχνει ο πρώτος δείκτης**.
  - Έχει **τιμή επιστροφής δείκτη σε χαρακτήρα**, όσο το **πρώτο της όρισμα**.
- **Αντιγραφή** μιας συμβολοσειράς σε άλλη γίνεται με την χρήση της συνάρτησης **strcpy( )**.
  - Η strcpy( ) δέχεται ως **ορίσματα δύο δείκτες σε χαρακτήρα**. **Αντιγράφει την συμβολοσειρά που ξεκινάει από εκεί που δείχνει ο δεύτερος δείκτης, εκεί που δείχνει ο πρώτος δείκτης**.
  - Έχει **τιμή επιστροφής δείκτη σε χαρακτήρα**, όσο το **πρώτο της όρισμα**.

Επαναλαμβάνεται μέρος της παραπάνω άσκησης 2, με καινούργια ζητούμενα:

Να γράψετε ένα πρόγραμμα, στο οποίο να περιγράψετε δύο είδη δομών, τις one και two. Οι δομές του είδους one έχουν δύο πεδία: το ak (ακέραιος) και το pin (πίνακας χαρακτήρων 30 θέσεων). Οι δομές του είδους two έχουν τέσσερα πεδία: το data (ακέραιος), το mat (πίνακας ακεραίων 5 θέσεων), το item (δομή του είδους one) και το melos (δομή του είδους one). Να δηλώσετε δυο μεταβλητές, την person (δομή του είδους one) και την memb (δομή του είδους two). Να δηλώσετε επίσης ένα δείκτη σε δομές του είδους one, τον ptr και ένα δείκτη σε δομές του είδους two, τον dkt. Να γράψετε τις εντολές (κάθε μια είναι ανεξάρτητη από τις υπόλοιπες), με τις οποίες θα γίνονται τα παρακάτω, μετά δε από κάθε εντολή να γράφονται στην οθόνη τα περιεχόμενα των δομών για επιβεβαίωση:

Όπου δηλώνεται πίνακας χαρακτήρων να υποθέσετε ότι αυτός περιέχει συμβολοσειρά.

Να κάνετε τον δείκτη ptr να δείχνει στην δομή person και τον dkt να δείχνει στην δομή memb και να γράψετε τις εντολές (κάθε μια είναι ανεξάρτητη από τις υπόλοιπες), με τις οποίες:

- Διαβάζουμε με την gets μια συμβολοσειρά για το πεδίο pin της δομής στην οποία δείχνει ο ptr.
- Αντιγράφουμε τη συμβολοσειρά που υπάρχει στον πίνακα pin της δομής στην οποία δείχνει ο ptr στον αντίστοιχο πίνακα του πεδίου item της της δομής στην οποία δείχνει ο dkt.
- Δίνουμε τιμή από το πληκτρολόγιο στο πεδίο data της δομής στην οποία δείχνει ο dkt.
- Δίνουμε τιμή στην δεύτερη θέση του πίνακα mat της δομής, στην οποία δείχνει ο dkt.
- Γράφουμε στην οθόνη τα περιεχόμενα των πεδίων της στην οποία δείχνει ο ptr.
- Αντιγράφουμε την δομή person, το item πεδό της δομής στην οποία δείχνει ο dkt.
- Αντιγράφουμε το πεδίο item της δομής στην οποία δείχνει ο dkt, στο πεδίο melos της ίδιας δομής.
- Γράφουμε στην οθόνη το μήκος της συμβολοσειράς που υπάρχει στον πίνακα χαρακτήρων του πεδίου melos της δομής στην οποία δείχνει ο dkt.

- Στο τέλος της συμβολοσειράς που υπάρχει στον πίνακα pin της person κάνουμε επικόλληση της συμβολοσειράς που υπάρχει στον αντίστοιχο πίνακα χαρακτήρων του πεδίου melos, της δομής στην οποία δείχνει ο dkt.

Επεξηγήσεις – υπενθυμίσεις (ασκήσεις 8, 9, 10, 11, 12, 13, 14) :

Ο τελεστής **sizeof δίνει το μέγεθος σε byte** του τελεστέου που βρίσκεται δεξιά του. Ο τελεστέος μπορεί να είναι **ένας προσδιοριστής τύπου σε παρενθέσεις**, όπως για παράδειγμα:

```
k = sizeof (float);
```

όπου ζητούμε από τον τελεστή sizeof **το μέγεθος σε byte των δεδομένων του είδους float**. Όταν αναφερόμαστε σε **συγκεκριμένη μεταβλητή** (και όχι γενικά σε είδος), της οποίας ζητούμε το μέγεθος σε byte, **η μεταβλητή αυτή δεν τίθεται σε παρενθέσεις**. Π.χ.:

```
float  fp;
int    ak;
ak = sizeof fp;
```

Δομές ως παράμετροι και ως τιμή επιστροφής συναρτήσεων.

8.

Επεξηγήσεις – υπενθυμίσεις (ασκήσεις 8, 10, 11, 12, 14) :

- Η συνάρτηση calloc( ) **δεσμεύει μνήμη** την ώρα της εκτέλεσης ενός προγράμματος.
- Δέχεται ως **ορίσματα δύο ακεραίους, τον αριθμό των στοιχείων μνήμης** που θα δεσμευτούν, καθώς και **το πλήθος των byte ανά στοιχείο μνήμης**.
- Εάν η **δέσμευση χώρου** είναι **επιτυχής, μηδενίζει τα περιεχόμενα της μνήμης** που δεσμεύει **και επιστρέφει ένα δείκτη στην αρχή** αυτού του χώρου.
- Η τιμή επιστροφής της είναι **δείκτης σε void** και γι'αυτό **χρειάζεται προσαρμογή τύπου.**
- Σε **αδυναμία δέσμευσης** μνήμης επιστρέφει **δείκτη ίσο με NULL.**

Να γραφεί ένα πρόγραμμα στο οποίο θα διαβάζετε ένα ακέραιο από το πληκτρολόγιο, τον num. Στη συνέχεια, με την χρήση της calloc( ) να δεσμεύσετε χώρο για να τοποθετήσετε num ακεραίους. Να εμφανίσετε τα περιεχόμενα του χώρου που δεσμεύτηκε στην οθόνη. Στη συνέχεια να διαβάσετε num ακέραιους, τους οποίους να τοποθετήσετε στον χώρο που δεσμεύτηκε και να εμφανίσετε ξανά τα περιεχόμενα του χώρου αυτού στην οθόνη.

Επεξηγήσεις – υπενθυμίσεις (ασκήσεις 9, 10, 11, 13, 14) :

- Η συνάρτηση malloc( ) **δεσμεύει μνήμη** την ώρα της εκτέλεσης ενός προγράμματος, της οποία μηδενίζει τα περιεχόμενα.
  - Δέχεται ως **όρισμα ένα ακέραιο, τον αριθμό των byte** που θέλουμε να δεσμεύσουμε.
  - Εάν η **δέσμευση χώρου** είναι **επιτυχής, επιστρέφει ένα δείκτη στην αρχή** αυτού του χώρου.
- Η συνάρτηση realloc( ) **τροποποιεί την ποσότητα μνήμης** που είχε προηγουμένως δεσμευτεί από κλήση της malloc( ) ή της calloc( ).
  - Δέχεται **δυο ορίσματα**: (α) Ένα **δείκτη στη θέση μνήμης**, της οποιας το μέγεθος θέλουμε να τροποποιήσουμε (δείκτης τον οποίο έχει **επιστρέψει μια κλήση της malloc( ) ή της calloc( )**). (β) Το **πλήθος των byte που θα δεσμευτούν**.
  - Αν τα byte που θα δεσμευτούν είναι **λιγότερα** από τα ήδη δεσμευμένα, τότε η realloc( ) δεσμεύει με **επιτυχία** τον χώρο.
  - Αν τα byte που θα δεσμευτούν είναι **περισσότερα** από τα ήδη δεσμευμένα και **υπάρχει ελεύθερος συνεχόμενος χώρος** μετά τον ήδη δεσμευμένο, η realloc( ) δεσμεύει τον χώρο που χρειάζεται επιπλέον.
  - Αν τα byte που θα δεσμευτούν είναι **περισσότερα** από τα ήδη δεσμευμένα και **δεν υπάρχει ελεύθερος συνεχόμενος χώρος** μετά τον ήδη δεσμευμένο, η realloc( ) **δεσμεύει χώρο σε νέα θέση**, τα υπάρχοντα δεδομένα **αντιγράφονται** στη νέα θέση, το παλιό μπλόκ μνήμης **απελευθερώνεται** και η συνάρτηση **επιστρέφει ένα δείκτη στην αρχή του νέου μπλόκ**.
- Και η malloc( ) και η realloc( ) σε **αδυναμία δέσμευσης** μνήμης επιστρέφουν **δείκτη ίσο με NULL**, ενώ η τιμή επιστροφής των είναι **δείκτης σε void** και γι'αυτό **χρειάζονται προσαρμογή τύπου.**

9.

Να διαβάσετε μια συμβολοσειρά από το πληκτρολόγιο, την pin. Να δεσμεύσετε με την malloc( ) όσο χώρο χρειάζεστε για την αποθήκευση της συμβολοσειράς αυτής. Στη συνέχεια να διαβάσετε μια άλλη συμβολοσειρά, την mat, την οποία να

αποθηκεύσετε στον χώρο της pin, μεγαλώνοντάς τον ή μικραίνοντάς τον με την χρήση της realloc( ). Στο τέλος να γράψετε στην οθόνη την pin.

Επεξηγήσεις – υπενθυμίσεις (ασκήσεις 10, 11) :

Με την δήλωση:

char *pin[N];

δημιουργούμε ένα **πίνακα δεικτών σε χαρακτήρες**, δηλαδή **ένα πίνακα σε κάθε θέση του οποίου υπάρχει ένας δείκτης σε χαρακτήρα**.

Άρα για παράδειγμα: το **pin[3]** είναι **δείκτης σε χαρακτήρα**, ενώ το ***pin[3]** είναι **χαρακτήρας.**

10.

Να γράψετε ένα πρόγραμμα, στο οποίο θα κάνετε τα εξής:

Θα δηλώσετε ένα πίνακα δεικτών σε χαρακτήρα MAX θέσεων. Στη συνέχεια να διαβάζετε συμβολοσειρές, για κάθε μια από τις οποίες να δεσμεύετε όσο χώρο χρειάζεται (με την malloc ή την calloc) και να τοποθετήσετε στην κάθε μία έναν από τους δείκτες του πίνακα δεικτών. Συμβολοσειρές να διαβάζετε μέχρι να διαβάσετε MAX συνολικά ή μέχρι να δώσετε κενή συμβολοσειρά.

Αφού τελειώσετε το διάβασμα, να εμφανισθούν στην οθόνη όλες οι συμβολοσειρές που διαβάσατε.

11. Τροποποίηση και επέκταση της ανωτέρω άσκησης 10.

Η εμφάνιση των συμβολοσειρών που διαβάσατε να γίνεται από μια συνάρτηση, την display( ), η οποία θα καλείται από την main( ).

12. Στο επόμενο παράδειγμα γίνεται δυναμική δημιουργία ενός διδιάστατου πίνακα ακεραίων nxm. Αφού το εκτελέσετε, να το συμπληρώσετε, ώστε να γράφονται στην οθόνη τα περιεχόμενα του πίνακα mat.

```
int main( ) {
    int k, j, n, m,  num, **mat;

    scanf ("%d%d", &n, &m);
    mat = (int **) malloc (n*sizeof (int *));

    for (k=0; k<n; k++)
        mat[k] = (int *) malloc (m*sizeof (int));
    for (j=0; j<n; j++) {
        for (k=0; k<m; k++) {
            printf ("mat[%d][%d]: \n", j, k);
            scanf ("%d", &num);
            mat[ j ][k] = num; } }
    return 1;}
```

13.

Επεξηγήσεις – υπενθυμίσεις (άσκηση 13) :

- **Επικόλληση** μιας συμβολοσειράς σε άλλη γίνεται με την χρήση της συνάρτησης **strcat( )**.
- Η strcat( ) δέχεται ως **ορίσματα δύο δείκτες σε χαρακτήρα**. **Επικολλά την συμβολοσειρά που ξεκινάει από εκεί που δείχνει ο δεύτερος δείκτης, στο τέλος της συμβολοσειράς που δείχνει ο πρώτος δείκτης**.
- Έχει **τιμή επιστροφής δείκτη σε χαρακτήρα**, όσο το **πρώτο της όρισμα**.

Να διαβάσετε μια συμβολοσειρά από το πληκτρολόγιο, την pin. Να δεσμεύσετε με την malloc( ) όσο χώρο χρειάζεστε για την αποθήκευση της συμβολοσειράς αυτής. Στη συνέχεια να διαβάσετε μια άλλη συμβολοσειρά, την mat. Να επικολλήσετε την mat στην pin, μεγαλώνοντας μικραίνοντας τον χώρο που κατείχε η pin με την χρήση της realloc( ). Στη συνέχεια να γράψετε στην οθόνη την pin.

14.

Επεξηγήσεις – υπενθυμίσεις (άσκηση 14) :

- **Αντιγραφή** μιας συμβολοσειράς σε άλλη γίνεται με την χρήση της συνάρτησης **strcpy( )**.
- Η strcpy( ) δέχεται ως **ορίσματα δύο δείκτες σε χαρακτήρα**. **Αντιγράφει την συμβολοσειρά που ξεκινάει από εκεί που δείχνει ο δεύτερος δείκτης, εκεί που δείχνει ο πρώτος δείκτης**.
- Έχει **τιμή επιστροφής δείκτη σε χαρακτήρα**, όσο το **πρώτο της όρισμα**.

Να γράψετε ένα πρόγραμμα, στο οποίο θα κάνετε τα εξής:

Να δηλώσετε ένα πίνακα δεικτών σε χαρακτήρα ΜΑΧ θέσεων. Στη συνέχεια να διαβάζετε συμβολοσειρές, για κάθε μια από τις οποίες να δεσμεύετε όσο χώρο χρειάζεται (με την malloc ή την calloc) και να τοποθετήσετε στην κάθε μία έναν από τους δείκτες του πίνακα δεικτών.

Αφού τελειώσετε το διάβασμα, να εμφανισετε στην οθόνη τη διεύθυνση του πρώτου χαρακτήρα κάθε συμβολοσειράς, καθώς και όλες τις συμβολοσειρές που διαβάσατε.

Στη συνέχεια, να αντιμεταθέσετε τιην πρώτη με την τελευταία συμβολοσειρά και να γράψετε ξανά όλες τις συμβολοσειρές στην οθόνη.

ΕΡΓΑΣΤΗΡΙΟ 3

Α’. ΠΕΡΙΕΧΟΜΕΝΑ ΜΑΘΗΜΑΤΟΣ

Στο εργαστήριο 3 καλύπτονται τα παρακάτω θέματα:

- Στοίβες**.** Υλοποίηση με πίνακα.
- Ουρές αναμονής και ουρές προτεραιότητας**.** Υλοποίηση με πίνακα.

Β’. ΑΣΚΗΣΕΙΣ ΓΙΑ ΕΚΤΕΛΕΣΗ – ΕΠΙΔΕΙΞΗ / ΕΠΕΞΗΓΗΣΕΙΣ

Στοίβες:

Επεξηγήσεις – υπενθυμίσεις (άσκηση 1) :

- Στην υλοποίηση στοίβας με πίνακα εξυπηρετεί η **δήλωση του πίνακα ως εξωτερικής μεταβλητής,** δεδομένου ότι μόνο δύο συναρτήσεις θα χειριστούν την στοίβα (ώθηση και ανάκληση).
- Το **σημείο εισόδου και εξόδου** (sp) στοιχείων στη στοίβα εξυπηρετεί επίσης **να δηλωθεί ως εξωτερική μεταβλητή** (π.χ. ακέραια μεταβλητή που θα ισούται με θέση του πίνακα).
- **Πριν την τοποθέτηση στοιχείου** στην στοίβα πρέπει να γίνεται **έλεγχος πλήρους στοίβας**, ενώ **πριν την ανάκληση στοιχείου** πρέπει να γίνεται **έλεγχος κενής στοίβας**.
- Η υλοποίηση της στοίβας μπορεί να γίνει με δύο λογικές:
    i) Εάν το sp δηλώνει την θέση της στοίβας όπου **θα τοποθετηθεί** στοιχείο, τότε κατά την **ώθηση τοποθετούμε στοιχείο στην στοίβα και αυξάνουμε το sp**, ενώ κατά την **ανάκληση ελαττώνουμε πρώτα το sp και μετά παίρνουμε το στοιχείο**.
    ii) Εάν το sp δηλώνει την θέση της στοίβας όπου **έχει τοποθετηθεί** το τελευταίο στοιχείο, τότε κατά την **ώθηση αυξάνουμε το sp και τοποθετούμε το στοιχείο στην στοίβα**, ενώ κατά την **ανάκληση παίρνουμε πρώτα το στοιχείο και μετά ελαττώνουμε το sp**.

1.

Να γραφεί ένα πρόγραμμα, στο οποίο να χρησιμοποιείτε μια στοίβα θετικών ακεραίων, την οποία να υλοποιήσετε με την χρήση πίνακα ακεραίων Ν θέσεων. Να γράψετε τις συναρτήσεις ώθησης (push) και ανάκλησης (pop) από την στοίβα, στις οποίες να γίνεται έλεγχος πλήρους και κενής στοίβας αντίστοιχα. Το πρόγραμμα να διαβάζει ένα χαρακτήρα από το πληκτρολόγιο, τον ch και:

- Εάν ο χαρακτήρας είναι το U να διαβάζετε ένα ακέραιο και να γίνεται ώθηση του ακεραίου στην στοίβα.
- Εάν ο χαρακτήρας είναι το Ο να γίνεται ανάκληση ακεραίου από την στοίβα, ο οποίος να εμφανίζεται στην οθόνη.
- Εάν ο χαρακτήρας είναι το Ε, να τελειώνει η όλη διαδικασία και να εμφανίζονται στην οθόνη τα περιεχόμενα της στοίβας.
- Εάν ο χαρακτήρας δεν είναι κανείς από τους παραπάνω, να γράφεται στην οθόνη «ΛΑΘΟΣ ΧΑΡΑΚΤΗΡΑΣ» και να διαβάζεται νέος χαρακτήρας από το πληκτρολόγιο.

Αφού τελειώσει η όλη διαδικασία, να γράφονται στην οθόνη οι ακέραιοι, οι οποίοι εξακολουθούν να υπάρχουν στην στοίβα.

Να υλοποιήσετε την στοίβα και με τις δύο λογικές που αναγράφονται πιο πάνω στις επεξηγήσεις.

Ουρές:

Επεξηγήσεις – υπενθυμίσεις (άσκηση 2) :

- Στην υλοποίηση ουράς με πίνακα εξυπηρετεί η **δήλωση του πίνακα ως εξωτερικής μεταβλητής,** δεδομένου ότι μόνο δύο συναρτήσεις θα χειριστούν την στοίβα (ώθηση και ανάκληση).
- Τα **σημεία εισόδου** (ip) **και εξόδου** (rp) στοιχείων στην ουρά εξυπηρετεί επίσης **να δηλωθούν ως εξωτερικές μεταβλητές** (π.χ. ακέραιες μεταβλητή που θα ισούνται με θέσεις του πίνακα).
- **Πριν την τοποθέτηση στοιχείου** στην ουρά πρέπει να γίνεται **έλεγχος πλήρους ουράς**, ενώ **πριν την ανάκληση στοιχείου** πρέπει να γίνεται **έλεγχος κενής ουράς**.
- **Κενή ουρά** έχουμε όταν **το ip συμπέσει με το rp.**

2.

Να γραφεί ένα πρόγραμμα, στο οποίο να υλοποιήσετε μια ουρά αναμονής θετικών ακεραίων, με την χρήση πίνακα ακεραίων Ν θέσεων. Να γράψετε τις συναρτήσεις ώθησης (qinsert) και ανάκλησης (qremove) από την ουρά, στις οποίες να γίνεται έλεγχος πλήρους και κενής ουράς αντίστοιχα. Το πρόγραμμα να διαβάζει ένα χαρακτήρα από το πληκτρολόγιο, τον ch και:

- Εάν ο χαρακτήρας είναι το U να διαβάζετε ένα ακέραιο και να γίνεται ώθηση του ακεραίου στην ουρά.
- Εάν ο χαρακτήρας είναι το Ο να γίνεται ανάκληση ακεραίου από την ουρά, ο οποίος να εμφανίζεται στην οθόνη.
- Εάν ο χαρακτήρας είναι το Ε, να τελειώνει η όλη διαδικασία και να εμφανίζονται στην οθόνη τα περιεχόμενα της ουράς.
- Εάν ο χαρακτήρας δεν είναι κανείς από τους παραπάνω, να γράφεται στην οθόνη «ΛΑΘΟΣ ΧΑΡΑΚΤΗΡΑΣ» και να διαβάζεται νέος χαρακτήρας από το πληκτρολόγιο.

Αφού τελειώσει η όλη διαδικασία, να γράφονται στην οθόνη οι ακέραιοι, οι οποίοι εξακολουθούν να υπάρχουν στην ουρά.

Να εμπλουτίσετε το πρόγραμμά σας με μια συνάρτηση, η οποία θα μηδενίζει τις τιμές των σημείων εισόδου και εξόδου στοιχείων στην περίπτωση που συμπέσουν (πράγμα που σημαίνει κενή ουρά).

3.

Επεξηγήσεις – υπενθυμίσεις (άσκηση 3) :

Ο **υπολογισμός του αποτελέσματος** με την χρήση στοίβας, όταν έχουμε **μεταθεματική παράσταση,** γίνεται με τα εξής βήματα:

- **Σαρώνουμε την παράσταση** από αριστερά προς τα δεξιά.
- Εάν συναντήσουμε **αριθμό**, τον **τοποθετούμε στην στοίβα**.
- Εάν συναντήσουμε **τελεστή**, **ανακαλούμε από την στοίβα τους τελεστέους στους οποίους αναφέρεται ο τελεστής, υπολογίζουμε το αποτέλεσμα και το ωθούμε στην στοίβα**.

Το τελικό αποτέλεσμα βρίσκεται στην κορυφή της στοίβας.

Να γραφεί ένα πρόγραμμα, το οποίο να υλοποιεί την μετατροπή από μεταδιατεταγμένη σε ενδοδιατεταγμένη παράσταση με τη χρήση στοίβας. Να διαβάζετε μια παράσταση με μορφή συμβολοσειράς, η οποία να περιέχει μονοψήφιους αριθμούς και τελεστές + ή – ή * ή /. Π.χ., την παράσταση:

5 9 4 + -7 3 * + +

Να υπολογίζεται η ένθετη μορφή (άρα για την παραπάνω παράσταση το αποτέλεσμα, το οποίο είναι -3).

4.

Επεξηγήσεις – υπενθυμίσεις (άσκηση 4) :

- Στις ουρές προτεραιότητας **ανακαλείται** κάθε φορά **το στοιχείο, το οποίο έχει τη μεγαλύτερη προτεραιότητα**. Είναι φανερό ότι **κάθε στοιχείο της ουράς** θα συνοδεύεται από **κάποια πρόσθετη πληροφορία, η οποία θα καθορίζει την προτεραιότητά του**, π.χ. μια ακέραια μεταβλητή
- **Αν η προτεραιότητα όλων των στοιχείων είναι η ίδια**, τότε **η λειτουργία είναι αυτή της κοινής ουράς**.

Να γραφεί ένα πρόγραμμα, στο οποίο να υλοποιείτε μια ουρά προτεραιότητας τριών επιπέδων (τριών βαθμών προτεραιότητας), με τη βοήθεια τριών πινάκων. Κάθε εισερχόμενο στοιχείο στην ουρά χαρακτηρίζεται από ένα όνομα και ένα αριθμό από 1 έως 3, ο οποίος δείχνει τον βαθμό προτεραιότητας του στοιχείου. Να χρησιμοποιήσετε μια δομή (structure) για το κάθε στοιχείο. Το πρόγραμμά σας να περιμένει να διαβάσει από το πληκτρολόγιο ένα χαρακτήρα. Εάν ο χαρακτήρας είναι το P να ζητείται το όνομα και ο βαθμός προτεραιότητας του στοιχείου και αυτό να τοποθετείται στην ουρά, στον κατάλληλο πίνακα. Εάν ο χαρακτήρας είναι το R να ανακαλείται το στοιχείο που πρέπει από την ουρά και να εμφανίζεται στην οθόνη. Εάν ο χαρακτήρας είναι το E, να τελειώνει η όλη διαδικασία και να εμφανίζονται στην οθόνη τα περιεχόμενα της ουράς, πρώτα αυτά με τη μεγαλύτερη προτεραιότητα, μετά με τη μεσαία και μετά με την χαμηλή.

ΕΡΓΑΣΤΗΡΙΟ 4

Α'. ΠΕΡΙΕΧΟΜΕΝΑ ΜΑΘΗΜΑΤΟΣ

Στο εργαστήριο 4 καλύπτονται τα παρακάτω θέματα:

- Απλά συνδεδεμένες λίστες. Δημιουργία.
- Λειτουργίες στις απλά συνδεδεμένες λίστες: αναζήτηση στοιχείου, εισαγωγή κόμβου, διαγραφή κόμβου, μετακίνηση κόμβου, αντιστροφή λίστας, συνένωση λιστών κλπ.

Β'. ΑΣΚΗΣΕΙΣ ΓΙΑ ΕΚΤΕΛΕΣΗ – ΕΠΙΔΕΙΞΗ / ΕΠΕΞΗΓΗΣΕΙΣ

Δείτε οπωσδήποτε την άσκηση 1 και να εκτελεστούν τουλάχιστον οι 2, 4 και 9.

Επεξηγήσεις – υπενθυμίσεις (όλες οι ασκήσεις) :

- Η δομή που περιγράφει κάθε κόμβο μιας απλά συνδεδεμένης λίστας περιέχει **«χρήσιμα» δεδομένα και ένα δείκτη προς ίδιου τύπου δομές**.
- Τον χώρο των κόμβων που δεν χρειαζόμαστε πρέπει να τον **ελευθερώνουμε** (συνάρτηση **free( )**).
- Στον **πρώτο κόμβο** (κεφαλή) μιας απλά συνδεδεμένης λίστας πρέπει να υπάρχει **ένας δείκτης, ο οποίος να μη μετακινηθεί ποτέ από εκεί**. Μέσω αυτού μπορούμε να διατρέχουμε την λίστα.
- Ο **δείκτης του τελευταίου κόμβου της λίστας** πρέπει να είναι **NULL**. Ελέγχοντας αν έχουμε φθάσει σε τιμή NULL, ξέρουμε αν φθάσαμε ή όχι στο τέλος της λίστας.

1. Το πρόγραμμα που ακολουθεί ζητεί να δημιουργηθεί μια απλά συνδεδεμένη λίστα. Τα στοιχεία της λίστας είναι του είδους node και περιέχουν το όνομα ενός ατόμου, τον αριθμό μητρώου του και το οφειλόμενο σε αυτόν ποσόν. Στοιχεία εισάγονται στην λίστα μέχρι να δοθεί αριθμός μητρώου μηδέν ή αρνητικός. Το πρόγραμμα δεν προβλέπει το ενδεχόμενο να δοθεί ίδιος αριθμός μητρώου για

δύο άτομα. Συνιστάται να το βελτιώσετε αργότερα (αφού κάνετε τις ασκήσεις), ώστε να αποκλείεται ένα τέτοιο ενδεχόμενο.

Μετά την καταχώρηση των στοιχείων, το πρόγραμμα, με τη χρήση της συνάρτησης display( ), εμφανίζει στην οθόνη τα ονόματα, τους αριθμούς μητρώου των ατόμων που έχουν καταχωρηθεί στην λίστα και το ποσόν που οφείλεται σε καθένα. Την συνάρτηση αυτή μπορείτε προφανώς να την χρησιμοποιείτε κάθε φορά που θέλετε να εμφανίσετε στην οθόνη τα περιεχόμενα της λίστας, δίνοντάς της ως όρισμα τον δείκτη στην κεφαλή της λίστας.

Όπως είναι διαρθωμένο το πρόγραμμα, δημιουργείται ένας κόμβος παραπάνω από όσους χρειαζόμαστε. Από την γραμμή 34 έως την γραμμή 37 του προγράμματος ελευθερώνουμε τον χώρο τον οποίο καταλαμβάνει ο κόμβος αυτός.

Μελετείστε το πρόγραμμα και αποθηκεύστε το, προκειμένου να εκτελέσετε και τις επόμενες ασκήσεις χρησιμοποιώντας το, χωρίς να ξαναδημιουργείτε την λίστα από την αρχή. Οι όποιες παρεμβάσεις σας προφανώς θα γίνονται αφού δημιουργηθεί η λίστα, άρα μετά την γραμμή 38 του προγράμματος.


```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>

struct node {
     char name [30];
     int   am;
     float poson;
     struct node *next; };


void display (struct node *);


main ( )
{
     struct node *head, *curr, *ptr;
     char nol[8];
     int num;

     /* Δημιουργία – Εισαγωγή στοιχείων στη λίστα */

     head = (struct node *) malloc (sizeof (struct node));
     curr = ptr = head;
     curr->next = NULL;
     printf ("DWSTE ARI8MO MHTRWOY ");
     gets (nol);
```

```
        num = atoi (nol);
        while (num > 0) {
                curr->am = num;
                printf ("DWSTE ONOMA ");
                gets (curr->name);
                printf ("DWSTE POSON ");
                gets (nol);
                curr->poson = atof (nol);
                curr->next = (struct node *) malloc (sizeof (struct node));
                curr = curr->next;
                curr->next = NULL;
                printf ("DWSTE EPOMENO ARI8MO MHTRWOY ");
                gets (nol);
                num = atoi (nol); }

        while (ptr->next != curr)                    /* Γραμμή 34 */
                ptr = ptr->next;
        ptr->next = NULL;
        free (curr);                                 /* Γραμμή 37 */
        curr = NULL;                                 /* Γραμμή 38 */
        display (head);
}

void display (struct node *head) {
        struct node *curr;

        /* Εμφάνιση των στοιχείων της λίστας στην οθόνη */

        printf ("\nEMFANISH TWN STOIXEIWN THS LISTAS\n");
        curr = head;
        while (curr != NULL) {
                printf ("ONOMA %s\t ARI8MOS MHTRWOY %5d\t POSON
                        %10.2f\n", curr->name, curr->am, curr->poson);
                curr = curr->next; }
}
```

2. Να γραφεί μία συνάρτηση, η ofeiles( ), η οποία να αθροίζει τα ποσά που οφείλονται σε όλα τα άτομα της λίστας της άσκησης 1 και να υπολογίζει τον μέσο όρο των οφειλόμενων ποσών στα άτομα της λίστας. Η main( ) να γράφει στην οθόνη το συνολικό οφειλόμενο ποσόν και τον μέσο όρο των οφειλομένων ποσών ανά άτομο.

3.

Επεξηγήσεις – υπενθυμίσεις (άσκηση 3) :

Η **σύγκριση συμβολοσειρών** γίνεται με την χρήση της **συνάρτησης strcmp( )**.

Να γραφεί μία συνάρτηση, η οποία να αναζητά ένα κόμβο στην λίστα της άσκησης 1, σύμφωνα με ένα όνομα που θα δίδεται στη main( ). Η συνάρτηση να επιστρέφει ένα δείκτη στον κόμβο που επισημάνθηκε το όνομα αυτό για πρώτη φορά. Εάν δεν υπάρχει το όνομα που ζητείται, η συνάρτηση να επιστρέφει NULL. Η main( ) να γράφει στην οθόνη το όνομα, τον αριθμό μητρώου του ατόμου και το ποσόν που οφείλεται σε αυτό (εφ' όσον το όνομα υπάρχει), αλλιώς να γράφει «ΑΝΥΠΑΡΚΤΟ ΟΝΟΜΑ».

4. Να γραφεί μία συνάρτηση, η οποία να κάνει εισαγωγή ενός κόμβου στην λίστα της άσκησης 1. Η συνάρτηση, εκτός από όσα άλλα ορίσματα χρειάζεται, να δέχεται ως παράμετρο και ένα ακέραιο, τον pos (η τιμή δηλαδή του pos θα διαβάζεται στην main και θα «στέλνεται» στην συνάρτηση). Η συνάρτηση θα δημιουργεί ένα καινούργιο κόμβο, θα του δίνει τιμές από το πληκτρολόγιο και θα κάνει τα εξής, ανάλογα με την τιμή του pos:

- Εάν ο pos είναι ίσος με μηδέν, να εισάγει τον καινούργιο κόμβο στην αρχή της υπάρχουσας λίστας.
- Εάν ο pos είναι ίσος με -1, να εισάγει τον καινούργιο κόμβο στο τέλος της υπάρχουσας λίστας.
- Εάν ο pos είναι ίσος με ένα θετικό ακέραιο, ας πούμε τον k, να εισάγει τον καινούργιο κόμβο μετά από τον k κόμβο της υπάρχουσας λίστας.
- Εάν ο pos είναι ίσος με αρνητικό αριθμό εκτός από -1, να ζητά να δοθεί από το πληκτρολόγιο ένας ακέραιος, ο num και να εισάγει τον καινούργιο κόμβο μετά από εκείνο τον κόμβο της υπάρχουσας λίστας, στον οποίο υπάρχει αριθμός μητρώου όσο το num.

Η συνάρτηση να επιστρέφει ένα δείκτη στην κεφαλή της λίστας, όπως έχει αυτή διαμορφωθεί μετά την εισαγωγή του νέου κόμβου. Με την χρήση της συνάρτησης display( ) της άσκησης 1 να εμφανίσετε στην οθόνη τα περιεχόμενα της λίστας μετά την εισαγωγή του νέου κόμβου.

5. Να γραφεί μία συνάρτηση, η οποία να κάνει διαγραφή ενός κόμβου από την λίστα της άσκησης 1. Η συνάρτηση, εκτός από όσα άλλα ορίσματα χρειάζεται, να δέχεται ως παράμετρο και ένα ακέραιο, τον pos (η τιμή δηλαδή του pos θα διαβάζεται στην main και θα «στέλνεται» στην συνάρτηση). Η συνάρτηση θα διαγράφει ένα κόμβο ως εξής, ανάλογα με την τιμή του pos:

- Εάν ο pos είναι ίσος με 0, να διαγράφει τον πρώτο κόμβο της λίστας.

- Εάν ο pos είναι ίσος με -1, να διαγράφει τον τελευταίο κόμβο της λίστας.
- Εάν ο pos είναι ίσος με ένας θετικό ακέραιο, ας πούμε τον k, να διαγράφει τον k κόμβο της λίστας.
- Εάν ο pos είναι ίσος με αρνητικό αριθμό εκτός από -1, να ζητά να δοθεί από το πληκτρολόγιο ένας ακέραιος, ο num και να διαγράφει τον κόμβο της λίστας, στον οποίο υπάρχει αριθμός μητρώου όσο το num.

Η συνάρτηση να επιστρέφει ένα δείκτη στην κεφαλή της λίστας, όπως έχει αυτή διαμορφωθεί μετά την διαγραφή του κόμβου. Με την χρήση της συνάρτησης display( ) της άσκησης 1 να εμφανίσετε στην οθόνη τα περιεχόμενα της λίστας μετά την διαγραφή του κόμβου.

6. Να δημιουργηθεί και μια δεύτερη λίστα όπως αυτή της άσκησης 1. Να γραφεί μία συνάρτηση, η οποία να συνενώνει τις δύο λίστες σε μία. Η main( ) να γράφει στην οθόνη τα περιεχόμενα της νέας ενιαίας λίστας.

7. Διαθέτετε μια απλά συνδεδεμένη λίστα. Να γράψετε μια συνάρτηση, η οποία θα μετατρέπει την λίστα σε κυκλική. Αυτό σημαίνει ότι ο δείκτης του τελευταίου κόμβου της λίστας δεν θα έχει τιμή ίση με NULL, αλλά θα δείχνει στον πρώτο κόμβο της λίστας. Η συνάρτηση να επιστρέφει ένα δείκτη στο μέγιστο στοιχείο της λίστας. Να θεωρήσετε ότι στην λίστα τοποθετούνται μόνο θετικοί ακέραιοι.

8.

Επεξηγήσεις – υπενθυμίσεις (άσκηση 8) :

- Η εισαγωγή ενός στοιχείου σε μια ουρά γίνεται με την **δημιουργία ενός νέου κόμβου και εισαγωγή** του στην λίστα. Η εισαγωγή μπορεί να γίνει **στο τέλος** ή **στην αρχή της υπάρχουσας λίστας**. Στην δεύτερη περίπτωση θυμηθείτε να μετακινήσετε τον δείκτη κεφαλής στην νέα κεφαλή της λίστας.
- Κατά την **εξαγωγή** στοιχείου, διαβάζουμε το στοιχείο από το άλλο άκρο της λίστας και **απελευθερώνουμε** (συνάρτηση free( )) τον αντίστοιχο κόμβο.
- **Πριν την τοποθέτηση στοιχείου** στην ουρά πρέπει να γίνεται **έλεγχος πλήρους ουράς**, (στην περίπτωσή μας δηλαδή εάν μπορεί να δεσμευθεί χώρος στην μνήμη για τον νέο κόμβο), ενώ **πριν την ανάκληση στοιχείου** πρέπει να γίνεται **έλεγχος κενής ουράς**.

Να υλοποιήσετε μια ουρά αναμονής με τη χρήση μιάς απλά συνδεδεμένης λίστας και τις συναρτήσεις εισαγωγής στοιχείου στην ουρά και διαγραφής στοιχείου από την ουρά. Να θεωρήσετε ότι στην ουρά τοποθετούνται μόνο θετικοί ακέραιοι.

9.

Επεξηγήσεις – υπενθυμίσεις (άσκηση 9) :

- Στην στοίβα **η θέση εισαγωγής** στοιχείων **συμπίπτει με την θέση εξαγωγής** στοιχείων (LIFO).
- Η εισαγωγή ενός νέου στοιχείου στην στοίβα γίνεται με την **δημιουργία ενός νέου κόμβου και εισαγωγή** του στην λίστα. Η εισαγωγή μπορεί να γίνει **στο τέλος** ή **στην αρχή της υπάρχουσας λίστας της υπάρχουσας λίστας**. Στην δεύτερη περίπτωση θυμηθείτε να μετακινήσετε τον δείκτη κεφαλής στην νέα κεφαλή της λίστας.
- Κατά την **εξαγωγή** στοιχείου, διαβάζουμε το στοιχείο από την λίστα και **απελευθερώνουμε** (συνάρτηση free( )) τον αντίστοιχο κόμβο.
- **Πριν την τοποθέτηση στοιχείου** στην στοίβα πρέπει να γίνεται **έλεγχος πλήρους στοίβας**, (στην περίπτωσή μας δηλαδή εάν μπορεί να δεσμευθεί χώρος στην μνήμη για τον νέο κόμβο), ενώ **πριν την ανάκληση στοιχείου** πρέπει να γίνεται **έλεγχος κενής στοίβας**.

Να υλοποιήσετε μια στοίβα με τη χρήση μιάς απλά συνδεδεμένης λίστας εισαγωγής στοιχείου στην στοίβα και διαγραφής στοιχείου από την στοίβα. Να θεωρήσετε ότι στην στοίβα τοποθετούνται μόνο θετικοί ακέραιοι.

10. Να γραφεί μία συνάρτηση, η οποία να αντιστρέφει την λίστα της άσκησης 1 και να επιστρέφει ένα δείκτη στην κεφαλή της νέας λίστας. Η αντιστροφή να γίνει χωρίς την δημιουργία νέας λίστας, αλλά με κατάλληλη αντιμετάθεση δεικτών (χρεάζεστε 3 βοηθητικούς δείκτες).

ΕΡΓΑΣΤΗΡΙΟ 5

Α’. ΠΕΡΙΕΧΟΜΕΝΑ ΜΑΘΗΜΑΤΟΣ

Στο εργαστήριο 5 καλύπτονται τα παρακάτω θέματα:

- Διπλά συνδεδεμένες λίστες. Δημιουργία.
- Λειτουργίες στις διπλά συνδεδεμένες λίστες: αναζήτηση στοιχείου, εισαγωγή κόμβου, διαγραφή κόμβου, μετακίνηση κόμβου, συνένωση λιστών κλπ.

Β’. ΑΣΚΗΣΕΙΣ ΓΙΑ ΕΚΤΕΛΕΣΗ – ΕΠΙΔΕΙΞΗ / ΕΠΕΞΗΓΗΣΕΙΣ

Δείτε οπωσδήποτε την άσκηση 1 και να εκτελεστούν τουλάχιστον οι 2, 5 και 9.

Επεξηγήσεις – υπενθυμίσεις (όλες οι ασκήσεις) :

- Η δομή που περιγράφει κάθε κόμβο μιας διπλά συνδεδεμένης λίστας περιέχει **«χρήσιμα» δεδομένα και δύο δείκτες προς ίδιου τύπου δομές**.
- Τον χώρο των κόμβων που δεν χρειαζόμαστε πρέπει να τον **ελευθερώνουμε** (συνάρτηση **free( )).**
- Σε **κάποιο κόμβο** μιας διπλά συνδεδεμένης λίστας πρέπει να υπάρχει **ένας δείκτης,** μέσω του οποίου μπορούμε να διατρέχουμε την λίστα. Τον δείκτη αυτόν προφανώς τον γνωρίζουμε, **δεν είναι όμως απαραίτητο να δείχνει στην άκρη της λίστας**. Αν θέλουμε κάτι τέτοιο, πρέπει να τον μετακινήσουμε εμείς.
- **Ο «αριστερός» δείκτης** του ενός άκρου της λίστας και **ο «δεξιός» δείκτης του άλλου άκρου** πρέπει να είναι **NULL**. Δηλαδή **η λίστα τερματίζεται προς δύο κατευθύνσεις**.

1. Το πρόγραμμα που ακολουθεί ζητεί να δημιουργηθεί μια διπλά συνδεδεμένη λίστα. Τα στοιχεία της λίστας είναι του είδους dnode και περιέχουν το όνομα ενός ατόμου, τον αριθμό μητρώου του και το οφειλόμενο σε αυτόν ποσόν. Στοιχεία εισάγονται στην λίστα μέχρι να δοθεί αριθμός μητρώου μικρότερος ή ίσος του μηδενός. Το πρόγραμμα δεν προβλέπει το ενδεχόμενο να δοθεί ίδιος αριθμός

μητρώου για δύο άτομα. Συνιστάται να το βελτιώσετε αργότερα (αφού κάνετε τις ασκήσεις), ώστε να αποκλείεται ένα τέτοιο ενδεχόμενο.

Όπως είναι διαρθωμένο το πρόγραμμα, δημιουργείται ένας κόμβος παραπάνω από όσους χρειαζόμαστε. Από την γραμμή 38 έως την γραμμή 40 του προγράμματος ελευθερώνουμε τον χώρο τον οποίο καταλαμβάνει ο κόμβος αυτός. Επίσης, δημιουργείται ένας δείκτης (ο one), ο οποίος είναι τοποθετημένος στον άκρο αριστερό κόμβο της λίστας.

Μετά την καταχώρηση των στοιχείων, το πρόγραμμα, με τη χρήση της συνάρτησης display1( ), εμφανίζει στην οθόνη τα ονόματα, τους αριθμούς μητρώου των ατόμων που έχουν καταχωρηθεί στην λίστα και το ποσόν που οφείλεται σε καθένα. Αυτό γίνεται για τους κόμβους της λίστας από εκεί που δείχνει το όρισμα της display1( ) και μετά (προς τα «δεξιά»).

Η συνάρτηση display2( ) δέχεται ως όρισμα ένα δείκτη σε κάποιο κόμβο της λίστας και εμφανίζει στην οθόνη τα ονόματα, τους αριθμούς μητρώου των ατόμων που έχουν καταχωρηθεί στην λίστα και το ποσόν που οφείλεται σε καθένα. Αυτό γίνεται για τους κόμβους της λίστας από το «δεξιό» της άκρο μέχρι το «άκρο αριστερό».

Τις συναρτήσεις display1( ) και display2( ) μπορείτε προφανώς να τις χρησιμοποιείτε κάθε φορά που θέλετε να εμφανίσετε στην οθόνη τα περιεχόμενα της λίστας, δίνοντας ως όρισμα κάθε φορά τον κατάλληλο δείκτη.

Μελετείστε το πρόγραμμα και αποθηκεύστε το, προκειμένου να εκτελέσετε και τις επόμενες ασκήσεις χρησιμοποιώντας το, χωρίς να ξαναδημιουργείτε την λίστα από την αρχή. Οι όποιες παρεμβάσεις σας προφανώς θα γίνονται αφού δημιουργηθεί η λίστα, άρα μετά την γραμμή 40 του προγράμματος.


```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>

struct dnode {
    char name[30];
    int  am;
    float poson;
    struct dnode *left;
    struct dnode *right; };

void display1 (struct dnode *);
void display2 (struct dnode *);
```

```
main( )
{
    struct dnode *one, *ptr, *two;
    int  num;
    char nol[8];

    /* Δημιουργία – εισαγωγή στοιχείων στην λίστα */

    one = (struct dnode *) malloc (sizeof (struct dnode));
    two = ptr = one;
    two->left = NULL;
    two->right = NULL;
    printf ("DWSTE ARI8MO MHTRWOY ");
    gets (nol);
    num = atoi (nol);
    while (num > 0) {
            two->am = num;
            printf ("DWSTE ONOMA ");
            gets (two->name);
            printf ("DWSTE POSON ");
            gets (nol);
            two->poson = atof (nol);
            two->right = (struct dnode *) malloc (sizeof (struct dnode));
            two->right->right = NULL;
            two->right->left = two;
            two = two->right;
            printf ("DWSTE EPOMENO ARI8MO MHTRWOY ");
            gets (nol);
            num = atoi (nol); }
    two->left->right = NULL;                          /* Γραμμή 38 */
    two->left = NULL;
    free (two);                                       /* Γραμμή 40 */

    printf ("\nEMFANISH STOIXEIWN THS LISTAS"
                                    " KATA THN ""OR8H"" FORA\n");
    display1 (one);
    printf ("\n");
    printf ("\nEMFANISH OLWN TWN STOIXEIWN THS LISTAS"
                                " KATA THN ""ANTISTROFH"" FORA\n");
    display2 (one); }

void display1 (struct dnode *ptr) {

    /* Εμφανίζει στην οθόνη τα στοιχεία της λίστας από εκεί που */
   /* δείχνει ο ptr και μετά, δηλαδή προς τα «δεξιά» */

    struct dnode *curr;
    curr = ptr;
    while (curr != NULL) {
    printf ("ONOMA %s\t ARI8MOS MHTRWOY %5d\t POSON %10.2f\n",
            curr->name, curr->am, curr->poson);
```

```
            curr = curr -> right; } }

void display2 (struct dnode *ptr) {
    /* Μετακινεί τον ptr στο «άκρο δεξιό» της λίστας και εμφανίζει */
   /* τα περιεχόμενά της προς τα «αριστερά» */

    struct dnode *curr;

    while (ptr->right != NULL)
            ptr = ptr -> right;
    curr = ptr;
    while (curr != NULL) {
    printf ("ONOMA %s\t ARI8MOS MHTRWOY %5d\t POSON %10.2f\n",
            curr->name, curr->am, curr->poson);
            curr = curr -> left; } }
```

2. Να γραφεί μία συνάρτηση, η ofeiles, η οποία να αθροίζει τα ποσά που οφείλονται σε όλα τα άτομα της λίστας της άσκησης 1 και να υπολογίζει τον μέσο όρο των οφειλόμενων ποσών στα άτομα της λίστας. Η main( ) να γράφει στην οθόνη το συνολικό οφειλόμενο ποσόν και τον μέσο όρο των οφειλομένων ποσών ανά άτομο.

Η συνάρτηση να θεωρείται προφανώς ότι «γνωρίζει» ένα δείκτη σε κάποιο κόμβο της λίστας.

3.

Επεξηγήσεις – υπενθυμίσεις (άσκηση 3) :

Η **σύγκριση συμβολοσειρών** γίνεται με την χρήση της **συνάρτησης strcmp( )**.

Να γραφεί μία συνάρτηση, η οποία να αναζητά ένα κόμβο στην λίστα της άσκησης 1, σύμφωνα με ένα όνομα που θα δίδεται. Η συνάρτηση να επιστρέφει ένα δείκτη στον κόμβο όπου επισημάνθηκε το όνομα για πρώτη φορά. Εάν δεν υπάρχει το όνομα που ζητείται, η συνάρτηση να επιστρέφει NULL. Η main( ) να γράφει στην οθόνη το όνομα που επεσήμανε, τον αριθμό μητρώου του συγκεκριμένου ατόμου και το ποσόν που οφείλεται στο άτομο αυτό. Η συνάρτηση να θεωρείται ότι «γνωρίζει» ένα δείκτη σε κάποιο κόμβο της λίστας.

4. Να γραφεί μία συνάρτηση, η οποία να κάνει εισαγωγή ενός κόμβου στην λίστα της άσκησης 1. Η συνάρτηση, εκτός από όσα άλλα ορίσματα χρειάζεται, να δέχεται ως όρισμα και ένα ακέραιο, τον pos (η τιμή δηλαδή του pos θα διαβάζεται στην main και θα «στέλνεται» στην συνάρτηση). Η συνάρτηση θα δημιουργεί ένα καινούργιο κόμβο, θα του δίνει τιμές από το πληκτρολόγιο και θα κάνει τα εξής, ανάλογα με την τιμή του pos:

- Εάν ο pos είναι ίσος με μηδέν, να εισάγει τον καινούργιο κόμβο στο ένα άκρο της υπάρχουσας λίστας.
- Εάν ο pos είναι ίσος με ένα θετικό ακέραιο, έστω τον k, να εισάγει τον καινούργιο κόμβο k κόμβους μετά από το ένα άκρο της υπάρχουσας λίστας.
- Εάν ο pos είναι ίσος με αρνητικό αριθμό, να ζητά να δοθεί από το πληκτρολόγιο ένας ακέραιος, ο num και να εισάγει τον καινούργιο κόμβο μετά από τον κόμβο της υπάρχουσας λίστας, στον οποίο υπάρχει αριθμός μητρώου όσο το num.

Η συνάρτηση να επιστρέφει ένα δείκτη στο ένα άκρο της λίστας, όπως έχει αυτή διαμορφωθεί μετά την εισαγωγή του νέου κόμβου.

5. Να γραφεί μία συνάρτηση, η οποία να κάνει διαγραφή ενός κόμβου από την λίστα της άσκησης 1. Η συνάρτηση, εκτός από όσα άλλα ορίσματα χρειάζεται, να δέχεται ως όρισμα και ένα ακέραιο, τον pos (η τιμή δηλαδή του pos θα διαβάζεται στην main και θα «στέλνεται» στην συνάρτηση). Η συνάρτηση θα διαγράφει ένα κόμβο ως εξής, ανάλογα με την τιμή του pos:

- Εάν ο pos είναι ίσος με μηδέν, να διαγράφει κόμβο από το ένα άκρο της λίστας.
- Εάν ο pos είναι ίσος με ένα θετικό ακέραιο, έστω τον k, να διαγράφει τον κόμβο που βρίσκεται k κόμβους μετά από το ένα άκρο της λίστας.
- Εάν ο pos είναι ίσος με αρνητικό αριθμό, να ζητά να δοθεί από το πληκτρολόγιο ένας ακέραιος, ο num και να διαγράφει τον κόμβο της λίστας, στον οποίο υπάρχει αριθμός μητρώου όσο το num.

Η συνάρτηση να επιστρέφει ένα δείκτη στο ένα άκρο της λίστας, όπως έχει αυτή διαμορφωθεί μετά την διαγραφή του κόμβου.

6. Να δημιουργηθεί και μια δεύτερη λίστα όπως αυτή της άσκησης 1. Να γραφεί μία συνάρτηση, η οποία να συνενώνει τις δύο λίστες σε μία. Η main( ) να γράφει στην οθόνη τα περιεχόμενα της νέας ενιαίας λίστας.

7. Διαθέτετε μια διπλά συνδεδεμένη λίστα. Να γράψετε μια συνάρτηση, η οποία θα μετατρέπει την λίστα σε κυκλική και θα επιστρέφει ένα δείκτη στο μέγιστο στοιχείο της λίστας. Να θεωρήσετε ότι στην λίστα τοποθετούνται μόνο θετικοί ακέραιοι.

8.

Επεξηγήσεις – υπενθυμίσεις (άσκηση 8) :

- Στην στοίβα **η θέση εισαγωγής** στοιχείων **συμπίπτει με την θέση εξαγωγής** στοιχείων (LIFO).
- Η εισαγωγή ενός νέου στοιχείου στην στοίβα γίνεται με την **δημιουργία ενός νέου κόμβου και εισαγωγή** του στην λίστα. Η εισαγωγή μπορεί να γίνει **στο τέλος της υπάρχουσας λίστας** ή **στην αρχή της υπάρχουσας λίστας**.
- Κατά την **εξαγωγή** στοιχείου, διαβάζουμε το στοιχείο από την λίστα και **απελευθερώνουμε** (συνάρτηση free( )) τον αντίστοιχο κόμβο.
- **Πριν την τοποθέτηση στοιχείου** στην στοίβα πρέπει να γίνεται **έλεγχος πλήρους στοίβας**, (στην περίπτωσή μας δηλαδή εάν μπορεί να δεσμευθεί χώρος στην μνήμη για τον νέο κόμβο), ενώ **πριν την ανάκληση στοιχείου** πρέπει να γίνεται **έλεγχος κενής στοίβας.**

Να υλοποιήσετε μια στοίβα με τη χρήση μιάς διπλά συνδεδεμένης λίστας. Να θεωρήσετε ότι στην στοίβα τοποθετούνται μόνο θετικοί ακέραιοι.

9.

Επεξηγήσεις – υπενθυμίσεις (άσκηση 9) :

- Στην ουρά αναμονής **η θέση εξαγωγής** στοιχείων «κυνηγάει» **την θέση εισαγωγής** στοιχείων (FIFO). **Όταν οι δύο θέσεις συμπέσουν**, η ουρά είναι **κενή**.
- Η εισαγωγή ενός νέου στοιχείου στην ουρά γίνεται με την **δημιουργία ενός νέου κόμβου και εισαγωγή** του στην λίστα. Η εισαγωγή μπορεί να γίνει **στο τέλος της υπάρχουσας λίστας** ή **στην αρχή της υπάρχουσας λίστας**.
- Κατά την **εξαγωγή** στοιχείου, διαβάζουμε το στοιχείο από το άλλο άκρο της λίστας και **απελευθερώνουμε** (συνάρτηση free( )) τον αντίστοιχο κόμβο.
- **Πριν την τοποθέτηση στοιχείου** στην ουρά πρέπει να γίνεται **έλεγχος πλήρους ουράς**, (στην περίπτωσή μας δηλαδή εάν μπορεί να δεσμευθεί χώρος στην μνήμη για τον νέο κόμβο), ενώ **πριν την ανάκληση στοιχείου** πρέπει να γίνεται **έλεγχος κενής ουράς**.

Να υλοποιήσετε μια ουρά αναμονής με τη χρήση μιάς διπλά συνδεδεμένης λίστας. Να θεωρήσετε ότι στην ουρά τοποθετούνται μόνο θετικοί ακέραιοι.

ΕΡΓΑΣΤΗΡΙΟ 6

Α’. ΠΕΡΙΕΧΟΜΕΝΑ ΜΑΘΗΜΑΤΟΣ

Στο εργαστήριο 6 καλύπτονται τα παρακάτω θέματα:

- Δέντρα. Δημιουργία δυαδικού δέντρου αναζήτησης (BST), διάσχιση.
- Λειτουργίες στα δυαδικά δέντρα αναζήτησης: αναζήτηση στοιχείου, εισαγωγή κόμβου, διαγραφή κόμβου κλπ.

Β’. ΑΣΚΗΣΕΙΣ ΓΙΑ ΕΚΤΕΛΕΣΗ – ΕΠΙΔΕΙΞΗ / ΕΠΕΞΗΓΗΣΕΙΣ

Δείτε οπωσδήποτε την άσκηση 1 και να εκτελεστούν τουλάχιστον οι 2, 3, 4 και 7.

Επεξηγήσεις – υπενθυμίσεις (όλες οι ασκήσεις) :

- Η δομή που περιγράφει κάθε κόμβο ενός δυαδικού δέντρου περιέχει **«χρήσιμα» δεδομένα και δύο δείκτες προς ίδιου τύπου δομές**.
- **Στην ρίζα** ενός δέντρου πρέπει να υπάρχει **ένας δείκτης,** μέσω του οποίου μπορούμε να διατρέχουμε το δέντρο. Ο δείκτης αυτός **δεν πρέπει να μετακινηθεί ποτέ από την ρίζα.**
- **Οι δείκτες των τελευταίων κόμβου του δέντρου** (φύλλων) πρέπει να είναι **NULL**.
- Ένα δυαδικό δέντρο αποτελεί **δέντρο αναζήτησης** εάν τα **«χρήσιμα» δεδομένα κάθε κόμβου του είναι μεγαλύτερα από τα δεδομένα όλων των αριστερών απογόνων του και μικρότερα από τα δεδομένα όλων των δεξιών απογόνων του**. Η σύγκριση γίνεται ως προς κάποιο από τα δεδομένα, ως προς κάποιο «κλειδί» δηλαδή.

1. Το πρόγραμμα που ακολουθεί δημιουργεί ένα δυαδικό δέντρο αναζήτησης. Τα στοιχεία του δέντρου είναι δομές του είδους ptr και περιέχουν ένα ακέραιο. Στοιχεία εισάγονται στο δέντρο μέχρι να δοθεί ακέραιος αρνητικός ή μηδέν.

Στη συνέχεια, διασχίζουμε το δέντρο που δημιουργήθηκε με τους τρείς τρόπους διάσχισης (ενδοδιατεταγμένο, προδιατεταγμένο και μεταδιατεταγμένο, γραμμές 33, 36 και 39 του προγράμματος) και εμφανίζουμε στην οθόνη τα αποτελέσματα. Με την συνάρτηση height( ) υπολογίζουμε το ύψος του δέντρου.

Στο πρόγραμμα, προκειμένου να διευκολυνθείτε στο γράψιμο των συναρτήσεών σας, γίνεται εμφάνιση των κόμβων του δέντρου και ανά επίπεδο (γραμμή 44). Σας δίνεται επίσης η συνάρτηση printGivenLevel ( ), η οποία σας επιτρέπει να γράψετε τα περιεχόμενα των κόμβων του δέντρου σε ένα συγκεκριμένο επίπεδο, το οποίο δίδεται ως όρισμα.

Μελετείστε το πρόγραμμα και αποθηκεύστε το, προκειμένου να εκτελέσετε και τις επόμενες ασκήσεις χρησιμοποιώντας το, χωρίς να ξαναδημιουργείτε το δέντρο από την αρχή. Δεν χρειάζεται να μελετήσετε με λεπτομέρειες τον τρόπο υλοποίησης του δέντρου. Θεωρείστε το ως ένα έτοιμο «εργαλείο», στο οποίο θα παρέμβετε με τις δικές σας συναρτήσεις. Μπορείτε λοιπόν να αφαιρέσετε από το πρόγραμμα τις συναρτήσεις που δεν χρειάζεστε ενώ δημιουργείτε τις δικές σας. Η δημιουργία του δέντρου έχει ολοκληρωθεί στην γραμμή 31.


```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct ptr {
     int  rec;
     struct  ptr *lp;
     struct  ptr *rp; } TREE;

TREE *treeLeaf (int);
TREE *newnode (int, TREE *, TREE *);

void inOrder (TREE *);
void preOrder (TREE *);
void postOrder (TREE *);
int height (TREE *);
void printLevelOrder (TREE *);
void printGivenLevel (TREE *, int);

main ( ) {
     TREE *root=(TREE *) NULL;
     char akin[5];
     int  ak, sum=0, hgt;

     gets (akin);
     ak = atoi (akin);
     root = newnode (ak, root, root);
     gets (akin);
     ak=atoi (akin);

     while (ak > 0) {
          newnode (ak, root, root);
          gets (akin);
          ak=atoi (akin); }                          /* Γραμμή 28 */
```

```
    puts ("Inorder");
    inOrder (root);                                   /* Γραμμή 30 */
    printf ("\n");
    puts ("Preorder");
    preOrder (root);                                  /* Γραμμή 33 */
    printf ("\n");
    puts ("Postorder");
    postOrder (root );                                /* Γραμμή 36 */
    printf ("\n");
    hgt = height (root);                              /* Γραμμή 38 */
    printf ("\nYPSOS TOY DENTROY = %5d\n", hgt);
    puts ("DIASXISH ANA EPIPEDO");
    printLevelOrder (root); }                         /* Γραμμή 41 */

TREE *treeLeaf (int num) {
    TREE *p;

    p = (TREE *) malloc (sizeof (TREE));
    p->rec = num;
    p->lp = NULL;
    p->rp = NULL;
    return (p); }

TREE *newnode (int num, TREE *pq, TREE *q) {
    if (q!=NULL) {
        if (num < q->rec)
            q = newnode (num, q, q->lp);
        else
            q = newnode (num, q, q->rp); }
    else {
        q = treeLeaf (num);
        if (pq!=NULL) {
            if (num < pq->rec)
                pq->lp = q;
            else
                pq->rp = q; }}
    return (q); }


/* Διάσχιση με ενδοδιατεταγμένο τρόπο */

void inOrder (TREE *q) {
    if (q!=NULL) {
        inOrder (q->lp);
        printf ("%d  ",q->rec);
        inOrder (q->rp); } }

/* Διάσχιση με προδιατεταγμένο τρόπο */

void preOrder (TREE *q) {
    if (q!=NULL) {
        printf ("%d  ", q->rec);
        preOrder (q->lp);
        preOrder (q->rp); } }
```

```
/* Διάσχιση με μεταδιατεταγμένο τρόπο */
void postOrder (TREE *q) {
      if (q!=NULL) {
           postOrder (q->lp);
           postOrder (q->rp);
           printf ("%d  ", q->rec); } }


/* Υπολογισμός ύψους δέντρου*/
int height(TREE* node) {
      int lheight, rheight;

      if (node==NULL)
            return 0;
       else {
            /* Υπολογισμός ύψους κάθε υποδέντρου */
            lheight = height (node->lp);
            rheight = height (node->rp);

            /* Χρησιμοποιούμε το μέγιστο από τα ύψη */
            if (lheight > rheight)
                 return (lheight+1);
            else
                 return (rheight+1); } }

/* Εμφάνιση κόμβων δέντρου ανά επίπεδο*/
void printLevelOrder (TREE* root) {
       int h = height (root);
       int i;
       for (i=1; i<=h; i++) {
            printGivenLevel (root, i);
            printf ("\n"); } }

/* Εμφάνιση κόμβων δέντρου σε ένα συγκεκριμένο επίπεδο */
void printGivenLevel (TREE* root, int level) {
      if (root == NULL)
            return;
      if (level == 1)
            printf ("%5d", root->rec);
      else
          if (level > 1) {
                printGivenLevel (root->lp, level-1);
                printGivenLevel (root->rp, level-1); } }
```


Επεξηγήσεις – υπενθυμίσεις (όλες οι ασκήσεις) :

Θυμηθείτε τις πιο πάνω συναρτήσεις **printLevelOrder** και **printGivenLevel.** Θα σας βοηθήσουν στα επόμενα.

2. Να δημιουργήσετε το δέντρο της άσκησης 1. Στη συνέχεια, να γράψετε μια συνάρτηση, η οποία θα αναζητά μέσα στο δέντρο ένα αριθμό που θα δίνετε από το πληκτρολόγιο (θυμηθείτε ότι έχουμε δυαδικό δέντρο αναζήτησης). Η συνάρτηση να επιστρέφει ένα δείκτη εκεί που επεσήμανε τον κόμβο ή NULL εάν ο αριθμός που αναζητούμε δεν υπάρχει στο δέντρο. Η main( ) να γράφει στην οθόνη τα παιδιά του κόμβου, τον οποίο επισημάνατε ή το μήνυμα «ΔΕΙΚΤΗΣ NULL», εάν ο αριθμός που ψάχνετε δεν υπάρχει στο δέντρο.

.

3. Να δημιουργήσετε το δέντρο της άσκησης 1 και να γράψετε μια συνάρτηση, η οποία να υλοποιεί τον αλγόριθμο εισαγωγής και να κάνει εισαγωγή ενός κόμβου στο δέντρο. Για παράδειγμα, σχεδιάστε ένα δέντρο, και υλοποιείστε το με τον κώδικα της άσκησης 1, δίνοντας κατά σειρά τις εξής τιμές στους κόμβους του δέντρου: 32 – 7 – 25 – 16 – 2 – 3 – 40 – 45 – 9 – 18 – 33 – 44 – 1 – 6. Στη συνέχεια, να καλέσετε την συνάρτησή σας και να κάνετε εισαγωγή της τιμής 26. Καλώντας μια από τις συναρτήσεις διάσχισης ή εμφάνισης κόμβων που δώσαμε παραπάνω, να διαπιστώσετε εάν ο κόμβος που εισαγάγατε μπήκε στη σωστή θέση.

4. Να δημιουργήσετε το δέντρο της άσκησης 1 και να γράψετε μια συνάρτηση, η οποία θα διαγράφει ένα κόμβο από το δέντρο. Για παράδειγμα, από το δέντρο που σχεδιάσατε στην παραπάνω άσκηση 3 να διαγράψετε την ρίζα. Καλώντας στη συνέχεια μια από τις συναρτήσεις διάσχισης ή εμφάνισης κόμβων που δώσαμε παραπάνω, να διαπιστώσετε εάν ο κόμβος που διγράψατε αφαιρέθηκε πράγματι από το δέντρο.

5. Να δημιουργήσετε το δέντρο της άσκησης 1. Να καλείτε μια συνάρτηση, η οποία να βρίσκει τον αδελφό κάποιου κόμβου του δέντρου. Η συνάρτηση να επιστρέφει ένα δείκτη στον κόμβο αυτόν, η δε main( ) να γράφει στην οθόνη τα δεδομένα που υπάρχουν στον κόμβο.

6. Να δημιουργήσετε το δέντρο της άσκησης 1. Να καλείτε μια συνάρτηση, η οποία να βρίσκει το αριστερότερο και το δεξιότερο φύλλο του δέντρου. Η main( ) να γράφει στην οθόνη τα δεδομένα των φύλλων αυτών.

7. Να δημιουργήσετε το δέντρο της άσκησης 1. Να καλείτε μια συνάρτηση, η οποία να διαγράφει τους κόμβους του δέντρου που περιέχουν το μικρότερο και το μεγαλύτερο στοιχείο. Μετά από τις διαγραφές, να εμφανίσετε το δέντρο στην οθόνη.

8. Να δημιουργήσετε το δέντρο της άσκησης 1. Να καλείτε την παρακάτω συνάρτηση, από την main( ), δίνοντάς της ως όρισμα το root. Η main( ) επίσης να γράφει στην οθόνη την τιμή επιστροφής της συνάρτησης. Μπορείτε να καταλάβετε τι κάνει;

```
int countL (TREE *x) {
    if (x == NULL) return 0;
    if (x->lp == NULL && x->rp == NULL) return 1;
    return countL(x->lp) + countL(x->rp); }
```

ΕΡΓΑΣΤΗΡΙΟ 7

Α’. ΠΕΡΙΕΧΟΜΕΝΑ ΜΑΘΗΜΑΤΟΣ

Στο εργαστήριο 7 καλύπτονται τα παρακάτω θέματα:

- Κατακερματισμός. Χρήση διαφόρων συναρτήσεων κατακερματισμού.
- Γράφοι.

Β’. ΑΣΚΗΣΕΙΣ ΓΙΑ ΕΚΤΕΛΕΣΗ – ΕΠΙΔΕΙΞΗ / ΕΠΕΞΗΓΗΣΕΙΣ

Να εκτελεστούν τουλάχιστον οι ασκήσεις 1, 3 και 6.

Κατακερματισμός.

Επεξηγήσεις – υπενθυμίσεις (ασκήσεις 1, 2) :

- Η συνάρτηση rand( ) δημιουργεί ένα τυχαίο θετικό ακέραιο. Πριν την πρώτη χρήση της πρέπει να κληθεί η συνάρτηση srand ως εξής: srand(time(NULL)).
- Στην **γραμμική δοκιμή** η δεύτερη (η δευτερεύουσα) συνάρτηση κατακερματισμού είναι η (h(k) + i) % m, με 1<= i <= n-1, όπου n το μέγεθος του πίνακα και m ακέραιος μικρότερος ή ίσος προς το μέγεθος του πίνακα.
- Στην **τετραγωνική δοκιμή** η δεύτερη (η δευτερεύουσα) συνάρτηση κατακερματισμού είναι η $(h(k) + c_1 * i + c_2 * i^2)$ % m, με 1<= i <= n-1, όπου n το μέγεθος του πίνακα, m ακέραιος μικρότερος ή ίσος προς το μέγεθος του πίνακα και c_1 και c_2 σταθερές.

1. Να γράψετε ένα πρόγραμμα, στο οποίο να δηλώσετε ένα πίνακα 31 θέσεων, τον pin. Να δημιουργήσετε 31 τυχαίους θετικούς ακέραιους μεταξύ 0 και 200, τους οποίους να τοποθετήσετε στον πίνακα εφαρμόζοντας αντιμετώπιση συγκρούσεων με γραμμική δοκιμή. Να εμφανίσετε στην οθόνη τον πίνακα pin μετά την τοποθέτηση των ακεραίων. Να θέσετε το m=31.

2. Να γράψετε ένα πρόγραμμα, στο οποίο να δηλώσετε ένα πίνακα 31 θέσεων, τον pin. Να δημιουργήσετε 31 τυχαίους θετικούς ακέραιους μεταξύ 0 και 200, τους οποίους να τοποθετήσετε στον πίνακα εφαρμόζοντας αντιμετώπιση συγκρούσεων με τετραγωνική δοκιμή. Να εμφανίσετε στην οθόνη τον πίνακα pin μετά την τοποθέτηση των ακεραίων. Να θέσετε τα m=31, c_1=1 και c_2 =2.

3. Να γράψετε ένα πρόγραμμα, στο οποίο να δηλώσετε ένα πίνακα 31 θέσεων, τον pin. Κάθε θέση του πίνακα θα περιέχει ένα δείκτη προς την κεφαλή μιας απλά συνδεδεμένης λίστας με δομές του είδους node, όπως η παρακάτω:

```
struct node {
    char name[20[;
    struct node *next; };
```

Ξεκινώντας, οι δείκτες αυτοί τίθενται όλοι ίσοι με NULL.

Να διαβάζετε συνεχώς συμβολοσειρές από το πληκτρολόγιο μέχρι να δώσετε κενή συμβολοσειρά. Κάθε συμβολοσειρά να υποθέσετε ότι έχει το πολύ 20 χαρακτήρες (μαζί με το \x0). Θα υπολογίζετε το μήκος της συμβολοσειράς (έστω ak) και θα τοποθετείτε την συμβολοσειρά σε μια από τις λίστες που ξεκινούν από τον pin. Η συμβολοσειρά θα τοποθετείται στο τέλος της ήδη υπάρχουσας λίστας. Η θέση του πίνακα, στην οποία θα τοποθετείται κάθε συμβολοσειρά θα προκύπτει από την χρήση μιας συνάρτησης κατακερματισμού, η οποία θα υπολογίζει το ak%31.

Όταν τελειώσετε με το διάβασμα συμβολοσειρών, να δώσετε ένα θετικό ακέραιο μικρότερο του 31, έστω τον num και να εμφανίζετε στην οθόνη τις συμβολοσειρές που έχουν αποθηκευτεί στην λίστα, στην οποία δείχνει ο num δείκτης του πίνακα δεικτών.

4. Το πρόγραμμα που ζητείται εδώ αποτελεί παραλλαγή της άσκησης 3. Συνιστάται να την επιχειρήσετε μόνο εάν θυμάστε πώς χειριζόμαστε αρχεία κειμένου.

Να γράψετε ένα πρόγραμμα, στο οποίο να δηλώσετε ένα πίνακα 101 θέσεων, τον pin. Κάθε θέση του πίνακα θα περιέχει ένα δείκτη προς την κεφαλή μιας απλά συνδεδεμένης λίστας με δομές του είδους node, όπως και στην άσκηση 3. Να διαβάζετε συνεχώς συμβολοσειρές από ένα αρχείο κειμένου, μέχρι να φτάσετε στο τέλος του αρχείου. Να αγνοήσετε τις συμβολοσειρές που έχουν περισσότερους από 20 χαρακτήρες (μαζί με το \x0). Να υπολογίζετε το μήκος κάθε συμβολοσειράς (έστω ak) και να την τοποθετείτε σε μια από τις λίστες που ξεκινούν από τον pin. Η συμβολοσειρά θα τοποθετείται στο τέλος της ήδη

υπάρχουσας λίστας. Η θέση του πίνακα, στην οποία θα τοποθετείται κάθε συμβολοσειρά θα προκύπτει από την χρήση μιας συνάρτησης κατακερματισμού, η οποία θα υπολογίζει το ak%101.

Στη συνέχεια να δώσετε μια λέξη από το πληκτρολόγιο και, χρησιμοποιώντας την ίδια συνάρτηση κατακερματισμού, να δείτε εάν υπάρχει στον πίνακα, άρα (έμμεσα) εάν υπάρχει στο αρχείο.

Γράφοι.

Επεξηγήσεις – υπενθυμίσεις (άσκηση 5) :

- Για ένα μη κατευθυνόμενο γράφο με n κορυφές ο **πίνακας γειτονικών κορυφών** είναι ένας n x n τετραγωνικός πίνακας, έστω pin, για τον οποίο:

  pin[j][k] = 1, αν η κορυφή v_j είναι γειτονική με την v_k και

  pin[j][k] = 0, αν η κορυφή v_j δεν είναι γειτονική με την v_k
- Εάν **πολλαπλασιάσουμε** τον πίνακα γειτονικών κορυφών **με τον εαυτό του**, προκύπτει ο **πίνακας μονοπατιών μήκους 2**.

5. Να σχεδιάσετε ένα μη κατευθυντικό γράφο 8 κορυφών της επιλογής σας. Να δώσετε τις εντολές από το πληκτρολόγιο για να δημιουργήσετε τον πίνακα γειτονικών κορυφών του γράφου. Να γράψετε μια συνάρτηση, η οποία να βρίσκει τον πίνακα μονοπατιών μήκους 2 του γράφου.

Η main( ) να γράφει στην οθόνη ποιες κορυφές του γράφου σας συνδέονται μεταξύ τους με μονοπάτια μήκους 2

Για διευκόλυνσή σας σας δίνεται ο κώδικας πολλαπλασιασμού μεταξύ τους δύο πινάκων NxN, των a και b, από τον οποίο πολλαπλασιασμό προκύπτει ο πίνακας c (διαστάσεων επίσης NxN):

```
for (j=0; j<N; j++)
    for (k=0; k<N; k++) {
        c[ j ][k] = 0;
        for (m=0; m<N; m++)
            c[ j ][k] = c[ j ][k] + a[ j ][m]*b[m][k]; }
```

Επεξηγήσεις – υπενθυμίσεις (άσκηση 6) :

- Στην παράσταση ενός γράφου με **λίστες γειτονικών κορυφών**, σε κάθε κορυφή του γράφου αντιστοιχούμε **μια απλά συνδεδεμένη λίστα**, η οποία περιέχει όλες τις γειτονικές κορυφές της αρχικής.
- Ο γράφος υλοποιείται με ένα **πίνακα δεικτών**, κάθε στοιχείο του οποίου είναι ένας δείκτης προς ένα κόμβο μιας λίστας. **Η κεφαλή της λίστας περιέχει τον αριθμό της κορυφής του γράφου και ένα δείκτη προς τον επόμενο κόμβο**, **ο οποίος** με την σειρά του **περιέχει τον αριθμό μιας κορυφής γειτονικής προς την αρχική και δείκτη προς τον επόμενο κόμβο** κοκ.

6. Στη main( ) ενός προγράμματος να διαβάσετε ένα ακέραιο, τον num, ο οποίος να αντιπροσωπεύει τις κορυφές ενός κατευθυντικού γράφου. Να σχεδιάσετε ένα κατευθυντικό γράφο και να γράψετε τις εξής συναρτήσεις:

- Την συνάρτηση insert( ), η οποία θα εισάγει μια κορυφή, ας πούμε την 1 στην λίστα της κορυφής 2 του πίνακα δεικτών (αυτό συμβαίνει εάν η 1 ενώνεται με την 2, μπορώ δηλαδή να μεταβώ από την 1 στην 2). Χρησιμοποιώντας τη συνάρτηση αυτή να δημιουργήσετε, να «χτίσετε» τον γράφο σας.
- Την συνάρτηση remove( ), η οποία θα «κόβει» την σύνδεση μιας κορυφής, με μια άλλη.
- Την συνάρτηση walk( ), η οποία θα δέχεται ως ορίσματα τρεις κορυφές, έστω τις 1, 2 και 3. Η συνάστηση θα δίνει αποτέλεσμα αληθές εάν μπορώ να μεταβώ από την 1 στην 2 μέσω της 3, αλλιώς θα δίνει αποτέλεσμα ψευδές (προφανώς αυτό να ισχύει για οποιεσδήποτε τρεις κορυφές δοθούν ως ορίσματα).