Ε.Α.Π./ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

4η ΓΡΑΠΤΗ ΕΡΓΑΣΙΑ

ΑΚΑΔΗΜΑΪΚΟΥ ΕΤΟΥΣ 2018-2019

3ος Τόμος

ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΕΠΙΣΤΗΜΗ ΤΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

5/3/2019

Ημερομηνία παράδοσης εργασίας: Κυριακή 05/05/2019
Καταληκτική ημερομηνία παραλαβής: Τετάρτη[1] 08/05/2019
Ημερομηνία ανάρτησης ενδεικτικών λύσεων: Σάββατο 11/05/2019
Καταληκτική ημερομηνία αποστολής σχολίων στον φοιτητή: Κυριακή 26/05/2019

ΥΠΟΕΡΓΑΣΙΑ 1. (βαθμοί 25)
Πίνακες δύο διαστάσεων

ΥΠΟΕΡΓΑΣΙΑ 2. (βαθμοί 25)
Συνδεδεμένες λίστες, Στοίβες

ΥΠΟΕΡΓΑΣΙΑ 3. (βαθμοί 25)
Διαπεράσεις δυαδικών δέντρων και δέντρα-σωροί

ΥΠΟΕΡΓΑΣΙΑ 4. (βαθμοί 25)
Συνδεδεμένες λίστες, Πίνακες μίας διαστάσεως

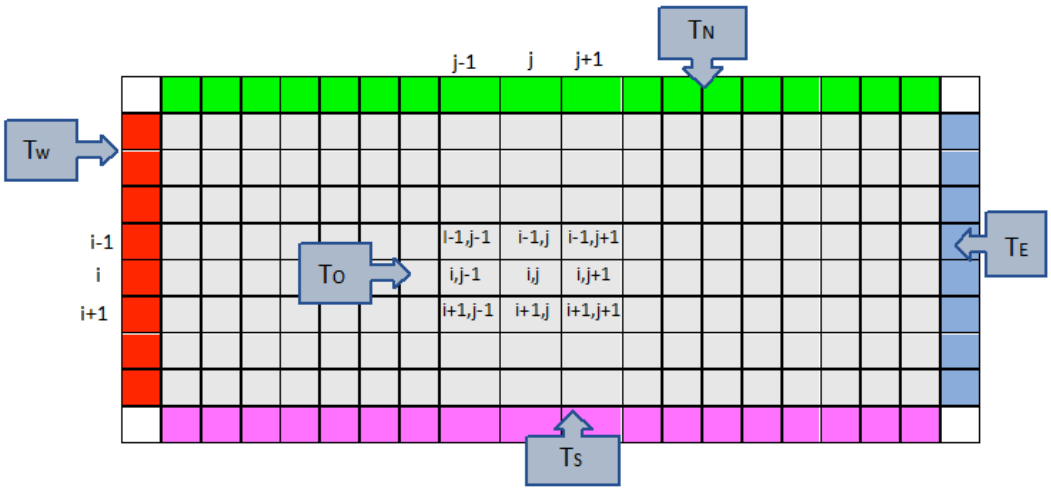
ΣΥΝΟΛΟ **(βαθμοί 100)**

[1] Σύμφωνα με τον Κανονισμό Σπουδών, η καταληκτική ημερομηνία για την παραλαβή της Γ.Ε. από το μέλος ΣΕΠ είναι η επόμενη Τετάρτη από το τέλος της εβδομάδας παράδοσης Γ.Ε.

ΥΠΟΕΡΓΑΣΙΑ 1 **(βαθμοί 25)**

Να δημιουργήσετε ένα πρόγραμμα στη γλώσσα προγραμματισμού C το οποίο να υπολογίζει τη θερμοκρασία σε βαθμούς Κελσίου σε διάφορα σημεία μιας μεταλλικής πλάκας από αγώγιμο υλικό. Στη μεταλλική πλάκα εφαρμόζονται διαφορετικές πηγές θερμότητας στις διάφορες πλευρές της. Το πρόγραμμα να υπολογίζει τη μετάδοση της θερμότητας στο αγώγιμο αυτό υλικό με την πάροδο του χρόνου και το χρόνο που απαιτείται για να μεταβεί η πλάκα στη «μόνιμη κατάσταση». Μόνιμη κατάσταση είναι η κατάσταση εκείνη, στην οποία καταλήγει η πλάκα *όταν η αθροιστική μεταβολή των θερμοκρασιών των στοιχείων της δεν υπερβαίνει τους 0.01 βαθμούς Κελσίου μετά την πάροδο ορισμένου χρονικού βήματος. Θεωρούμε ότι το χρονικό βήμα είναι ένα δευτερόλεπτο* και ότι η μεταβολή θερμοκρασίας σε ένα στοιχείο υπολογίζεται ως *η απόλυτη τιμή* της διαφοράς των διαδοχικών τιμών του.

Η πλάκα χωρίζεται σε *DIMY* επί *DIMX* τετραγωνικά στοιχεία, (έστω *DIMY*=10, *DIMX*=20, όπως φαίνεται στο σχήμα). Υποθέτουμε ότι η θερμοκρασία στις τέσσερεις πλευρές της πλάκας ***παραμένει σταθερή*** καθόλη τη διάρκεια του φαινομένου**.** ***Στα τέσσερα γωνιακά στοιχεία*** η θερμοκρασία *υπολογίζεται ως η μέση τιμή των θερμοκρασιών των προσκείμενων πλευρών*. Το πρόγραμμα θα πρέπει να ζητάει από τον χρήστη να ορίσει τις θερμοκρασίες των τεσσάρων πλευρών με τις μεταβλητές T_W, T_E, T_N, T_S που αναφέρονται στις πλευρές της πλάκας στα τέσσερα σημεία του χάρτη (*West* (Δυτικά), *East* (Ανατολικά), *North* (Βόρεια), *South* (Νότια)) και με βάση τις τιμές αυτές να προσδιορίζει τις τιμές θερμοκρασίας των γωνιακών στοιχείων. ***Τα υπόλοιπα στοιχεία*** της πλάκας έχουν ***όλα*** τη χρονική στιγμή 0 την αρχική θερμοκρασία της πλάκας που επίσης ζητείται από τον χρήστη και συμβολίζεται με T_0 (βλέπε σχήμα).



Ο υπολογισμός της θερμοκρασίας κάθε στοιχείου της πλάκας σε κάθε επόμενη χρονική στιγμή *t* δίνεται ως συνάρτηση των θερμοκρασιών του ίδιου και των οκτώ γειτονικών στοιχείων του κατά την αμέσως προηγούμενη χρονική στιγμή *t-1*.

Ο τύπος υπολογισμού της θερμοκρασίας $T_{t(i,j)}$ του στοιχείου *(i,j)* κατά τη χρονική στιγμή *t* δίνεται από τον παρακάτω τύπο:

$$T_{t(i,j)} = 0.1 * (T_{t-1(i-1,j-1)} + T_{t-1(i-1,j)} + T_{t-1(i-1,j+1)} + T_{t-1(i,j-1)} + 2 * T_{t-1(i,j)} + T_{t-1(i,j+1)} + T_{t-1(i+1,j-1)} + T_{t-1(i+1,j)} + T_{t-1(i+1,j+1)})$$

Ζητούνται τα εξής:

Α. Να ορίσετε συνάρτηση με όνομα `float EnterTemperature(const char prompt[51])` που να ζητεί από τον χρήστη μία θερμοκρασία. Η συνάρτηση να δέχεται ως όρισμα μήνυμα προς τον χρήστη, δηλαδή ένα αλφαριθμητικό μήκους το πολύ 50 χαρακτήρων που να περιγράφει τι ζητείται. Η συνάρτηση να επιστρέφει την τιμή που εισάγει ο χρήστης.

Β. Να ορίσετε μια συνάρτηση με όνομα `void InitializeTable(float T[DIMY][DIMX])` η οποία να αρχικοποιεί ένα πίνακα πραγματικών αριθμών με όνομα *T* διαστάσεων 10 *x* 20 στοιχείων (*DIMY*=10, *DIMX*=20) που να αναπαριστά τις θερμοκρασίες των στοιχείων μιας μεταλλικής πλάκας, με βάση τις τιμές θερμοκρασιών που δίνει ο χρήστης. Για να πραγματοποιηθεί η αρχικοποίηση, η συνάρτηση ζητάει από τον χρήστη να δώσει τη θερμοκρασία των διαφόρων στοιχείων, όπως αναφέρθηκε παραπάνω (θερμοκρασίες T_W, T_E, T_N, T_S και T_0) καλώντας τη συνάρτηση `EnterTemperature` περνώντας το κατάλληλο για κάθε θερμοκρασία μήνυμα στο χρήστη ως όρισμα.

Γ. Να ορίσετε μια συνάρτηση με όνομα `void PrintTable(float T[DIMY][DIMX])` η οποία να τυπώνει τα περιεχόμενα του πίνακα *T*. Η εκτύπωση των στοιχείων του πίνακα να γίνεται με δύο δεκαδικά ψηφία και αλλάζοντας γραμμή εκτύπωσης κατά την αλλαγή γραμμής του πίνακα, όπως δείχνει το απόσπασμα εκτέλεσης που ακολουθεί.

Δ. Να ορίσετε μια συνάρτηση με όνομα `float CalculateNextTemperature(float T[DIMY][DIMX], float C[DIMY][DIMX])` η οποία να υπολογίζει τη νέα θερμοκρασία των στοιχείων της πλάκας σύμφωνα με τον δοθέντα τύπο υπολογισμού μεταβολής της θερμοκρασίας. Το όρισμα *C* αναπαριστά τη μεταλλική πλάκα τη χρονική στιγμή *t-1* και το όρισμα *T την ίδια* μεταλλική πλάκα τη χρονική στιγμή *t*. Η συνάρτηση να επιστρέφει την *αθροιστική μεταβολή θερμοκρασιών όλων των στοιχείων της πλάκας.*

Ε. Να ορίσετε τη συνάρτηση `main` η οποία να δηλώνει ως τοπική μεταβλητή έναν πίνακα `Plate` διαστάσεων 10 *x* 20 στοιχείων (*DIMY*=10, *DIMX*=20), που να αναπαριστά την μεταλλική πλάκα. Η `main` αρχικά θα πρέπει να καλεί τη συνάρτηση `InitializeTable` για να αρχικοποιήσει τον πίνακα `Plate`, στη συνέχεια να καλεί επαναληπτικά τη συνάρτηση `CalculateNextTemperature` για να υπολογίσει τη νέα κατάσταση της πλάκας και *την*

αθροιστική μεταβολή των θερμοκρασιών των στοιχείων της και να τυπώνει, με κλήση της συνάρτησης `PrintTable`, την κατάσταση της μεταλλικής πλάκας *σε κάθε επανάληψη*. Ο επαναληπτικός αυτός υπολογισμός θα πρέπει να τερματιστεί όταν η πλάκα έχει φτάσει σε μόνιμη κατάσταση, δηλαδή όταν *η αθροιστική μεταβολή των θερμοκρασιών των στοιχείων της δεν υπερβαίνει τους 0.01 βαθμούς Κελσίου*. Το πρόγραμμα θα πρέπει να υπολογίζει το χρόνο σε δευτερόλεπτα, που απαιτήθηκε για την μετάβαση της πλάκας στη μόνιμη κατάσταση για τις δεδομένες αρχικές συνθήκες και πριν τερματίσει να τον εκτυπώνει.

Ενδεικτικά δίνεται ένα απόσπασμα εκτέλεσης με βάση την περιγραφή που δόθηκε, στο οποίο φαίνονται οι αρχικές τιμές θερμοκρασίας της μεταλλικής πλάκας (*t=0*) και οι τιμές της θερμοκρασίας της πλάκας κατά τη χρονική στιγμή *t=1* δευτερόλεπτο.

```
Εισαγωγή Θερμοκρασίας T West :  4
Εισαγωγή Θερμοκρασίας T East : -5
Εισαγωγή Θερμοκρασίας T North : 2
Εισαγωγή Θερμοκρασίας T South : 3
Εισαγωγή Θερμοκρασίας Πλάκας T(0) : 1
T0
 3.00  2.00  2.00  2.00  2.00  2.00  2.00  2.00  2.00  2.00  2.00  2.00  2.00  2.00  2.00  2.00  2.00  2.00  2.00 -1.50
 4.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00 -5.00
 4.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00 -5.00
 4.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00 -5.00
 4.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00 -5.00
 4.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00 -5.00
 4.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00 -5.00
 4.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00 -5.00
 4.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00 -5.00
 3.50  3.00  3.00  3.00  3.00  3.00  3.00  3.00  3.00  3.00  3.00  3.00  3.00  3.00  3.00  3.00  3.00  3.00  3.00 -1.00

T1
 3.00  2.00  2.00  2.00  2.00  2.00  2.00  2.00  2.00  2.00  2.00  2.00  2.00  2.00  2.00  2.00  2.00  2.00  2.00 -1.50
 4.00  2.00  1.30  1.30  1.30  1.30  1.30  1.30  1.30  1.30  1.30  1.30  1.30  1.30  1.30  1.30  1.30  1.30 -0.25 -5.00
 4.00  1.90  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00 -0.80 -5.00
 4.00  1.90  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00 -0.80 -5.00
 4.00  1.90  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00 -0.80 -5.00
 4.00  1.90  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00 -0.80 -5.00
 4.00  1.90  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00 -0.80 -5.00
 4.00  1.90  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00  1.00 -0.80 -5.00
 4.00  2.25  1.60  1.60  1.60  1.60  1.60  1.60  1.60  1.60  1.60  1.60  1.60  1.60  1.60  1.60  1.60  1.60  0.00 -5.00
 3.50  3.00  3.00  3.00  3.00  3.00  3.00  3.00  3.00  3.00  3.00  3.00  3.00  3.00  3.00  3.00  3.00  3.00  3.00 -1.00
```

Στο σχέδιο προγράμματος με τίτλο `ypoergasia1_code_template.c` καλείστε να συμπληρώσετε τον κώδικα των συναρτήσεων που περιγράφηκαν, *ενώ μπορείτε να ορίσετε και δικές σας βοηθητικές συναρτήσεις.*

ΥΠΟΕΡΓΑΣΙΑ 2 (βαθμοί 25)

Γράψτε ένα πρόγραμμα σε γλώσσα προγραμματισμού C το οποίο να δημιουργεί μια στοίβα όπου θα καταχωρίζεται ο ***Βαθμός*** στη Θεματική Ενότητα ΠΛΗ10 για κάθε φοιτητή του ΕΑΠ. Η υλοποίηση της στοίβας θα πρέπει να γίνει με *απλά συνδεδεμένη* λίστα. Να ζητείται από τον χρήστη να εισάγει την βαθμολογία φοιτητή/τριας στην ΘΕ ΠΛΗ10 του ΕΑΠ. Η βαθμολογία πρέπει να παίρνει ακέραιες τιμές στο κλειστό διάστημα [1,10]. Με χρήση αμυντικού προγραμματισμού να εξασφαλίζεται ότι οι τιμές που εισάγονται θα είναι εντός των αποδεκτών ορίων. *Θεωρούμε ως δεδομένο ότι η βαθμολογία παίρνει ακέραιες τιμές.*

Ο χρήστης να ερωτάται να επιλέξει ανάμεσα σε δύο επιλογές. Αν θέλει να εισάγει νέα βαθμολογία να του δίνεται αυτή η δυνατότητα εισάγοντας 1 από το πληκτρολόγιο. Αν εισαχθεί το 0 η εισαγωγή να τερματίζεται. Με χρήση αμυντικού προγραμματισμού να γίνονται αποδεκτές μόνο οι τιμές 0 και 1. Όλες οι αποδεκτές τιμές βαθμολογίας να καταχωρίζονται στη στοίβα (με βάση τις ιδιότητες της στοίβας) και να εμφανίζονται μία προς μία *από την κορυφή προς τα κάτω*. Με χρήση αμυντικού προγραμματισμού να πραγματοποιείται έλεγχος ύπαρξης διαθέσιμης μνήμη κατά την καταχώριση ενός νέου κόμβου εμφανίζοντας αντίστοιχο μήνυμα σε περίπτωση αποτυχίας της δυναμικής δέσμευσης μνήμης. Στη συνέχεια η στοίβα να ***ταξινομείται*** σε φθίνουσα διάταξη και να εμφανίζονται τα στοιχεία της ***ταξινομημένα***.

Το πρόγραμμα πρέπει να περιέχει τις ακόλουθες συναρτήσεις:

Α. Τη συνάρτηση `void construct(struct stack **head)` όπου `head` είναι *διπλός* δείκτης (δείκτης σε δείκτη) για την προσπέλαση στην κεφαλή της στοίβας. Η συνάρτηση αρχικοποιεί μια άδεια στοίβα.

Β. Τη συνάρτηση `int isEmpty(struct stack *head)` όπου `head` είναι απλός δείκτης για την προσπέλαση στην κεφαλή της στοίβας. Η συνάρτηση ελέγχει αν η στοίβα είναι άδεια.

Γ. Τη συνάρτηση `void push(struct stack **head, int x)` η οποία εισάγει την τιμή `x` στη στοίβα αφού δεσμεύσει δυναμικά τον απαιτούμενο χώρο μνήμης.

Δ. Τη συνάρτηση `int pop(struct stack **head)` η οποία εξάγει το κορυφαίο στοιχείο από τη στοίβα και επιστρέφει την τιμή του.

Ε. Τη συνάρτηση `int top(struct stack *head)` η οποία επιστρέφει το στοιχείο της κορυφής της στοίβας.

ΣΤ. Την αναδρομική συνάρτηση `void Insert_Sort(struct stack **head, int vath)` η οποία εισάγει το στοιχείο `vath` μέσα στη στοίβα έτσι ώστε μετά από κάθε εισαγωγή η στοίβα να είναι ταξινομημένη. Θέλουμε η ταξινόμηση να είναι σε *φθίνουσα* διάταξη βαθμολογίας.

Ζ. Την αναδρομική συνάρτηση `void sort_Stack(struct stack **head)` η οποία υλοποιεί την διαδικασία ταξινόμησης καλώντας την συνάρτηση `Insert_Sort`.

Η. Την αναδρομική συνάρτηση `void show(struct node *head)` η οποία εμφανίζει όλα τα στοιχεία της στοίβας. Η συνάρτηση καλείται να εμφανίσει πρώτα την αρχική μορφή της στοίβας και μετά τη μορφή της στοίβας που προκύπτει μετά την ταξινόμηση.

Θ. Τη συνάρτηση `void st_delete(struct node **head)` η οποία διαγράφει τη στοίβα από την μνήμη την οποία και απελευθερώνει.

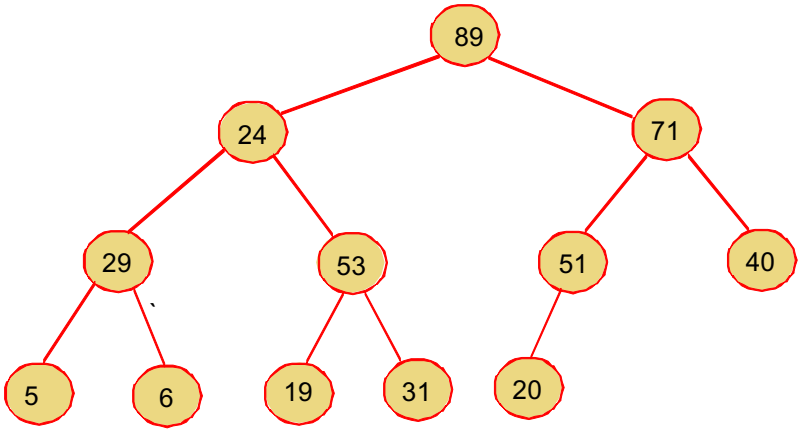
Ακολουθεί εικόνα με παράδειγμα εκτέλεσης του προγράμματος:

```
Σύστημα καταχώρησης βαθμολογίας φοιτητών ΕΑΠ στην ΘΕ ΠΛΗ10
**********************************************************


Δώστε τον βαθμό : 12
Η βαθμολογία πρέπει να είναι από 1 έως και 10. Λάθος εισαγωγή τιμής
Η βαθμολογία που δώσατε δεν θα καταχωρηθεί

Παρακαλώ Ξαναδώστε τον βαθμό :6

Ο βαθμός που δώσατε έχει καταχωρηθεί.

Θα δώσετε και άλλον βαθμό; (1=ΝΑΙ,0=ΟΧΙ)1

Δώστε τον βαθμό : 9
Ο βαθμός που δώσατε έχει καταχωρηθεί.

Θα δώσετε και άλλον βαθμό; (1=ΝΑΙ,0=ΟΧΙ)1

Δώστε τον βαθμό : 8
Ο βαθμός που δώσατε έχει καταχωρηθεί.

Θα δώσετε και άλλον βαθμό; (1=ΝΑΙ,0=ΟΧΙ)1

Δώστε τον βαθμό : 10
Ο βαθμός που δώσατε έχει καταχωρηθεί.

Θα δώσετε και άλλον βαθμό; (1=ΝΑΙ,0=ΟΧΙ)1

Δώστε τον βαθμό : 3
Ο βαθμός που δώσατε έχει καταχωρηθεί.

Θα δώσετε και άλλον βαθμό; (1=ΝΑΙ,0=ΟΧΙ)1

Δώστε τον βαθμό : 5
Ο βαθμός που δώσατε έχει καταχωρηθεί.

Θα δώσετε και άλλον βαθμό; (1=ΝΑΙ,0=ΟΧΙ)1

Δώστε τον βαθμό : 1
Ο βαθμός που δώσατε έχει καταχωρηθεί.

Θα δώσετε και άλλον βαθμό; (1=ΝΑΙ,0=ΟΧΙ)0
Οι βαθμοί της στοίβας πριν την ταξινόμηση είναι:
Ο βαθμός του φοιτητή είναι 1
Ο βαθμός του φοιτητή είναι 5
Ο βαθμός του φοιτητή είναι 3
Ο βαθμός του φοιτητή είναι 10
Ο βαθμός του φοιτητή είναι 8
Ο βαθμός του φοιτητή είναι 9
Ο βαθμός του φοιτητή είναι 6


Οι βαθμοί της στοίβας μετά την ταξινόμηση είναι:
Ο βαθμός του φοιτητή είναι 10
Ο βαθμός του φοιτητή είναι 9
Ο βαθμός του φοιτητή είναι 8
Ο βαθμός του φοιτητή είναι 6
Ο βαθμός του φοιτητή είναι 5
Ο βαθμός του φοιτητή είναι 3
Ο βαθμός του φοιτητή είναι 1
```

Θα πρέπει να χρησιμοποιήσετε υποχρεωτικά το σχέδιο προγράμματος με όνομα αρχείου `ypoergasia2_code_template.c`. Στο πρότυπο αυτό οι συναρτήσεις `construct`, `is_Empty`, `push`, `pop`, `top`, `st_delete`, και τμήματα των υπολοίπων συναρτήσεων είναι ήδη υλοποιημένα και πρέπει να συμπληρώσετε τα τμήματα που λείπουν.

ΥΠΟΕΡΓΑΣΙΑ 3 (βαθμοί 25)

Α. Δίνεται το εξής δυαδικό δέντρο:



i) Εξετάστε αν το δέντρο αυτό είναι *δέντρο-σωρός μεγίστων*. Αιτιολογήστε την απάντησή σας.

ii) Εισάγετε τα δεδομένα των κόμβων του δέντρου αυτού με τη σειρά που εμφανίζονται κατά την *ενδοδιατεταγμένη* διαπέραση του σε ένα άλλο, αρχικά κενό, δέντρο-σωρό μεγίστων.

Να δείξετε το δέντρο-σωρό μεγίστων που σχηματίζεται μετά την εισαγωγή των στοιχείων 24, 31, 51, 40.

iii) Στο τελικό δέντρο-σωρό μεγίστων που προκύπτει μετά την εισαγωγή όλων των δεδομένων του υποερωτήματος ii να γίνει *διαγραφή* της ρίζας. Δώστε το δέντρο-σωρό μεγίστων που προκύπτει από τη διαγραφή αυτή μαζί με μία σύντομη αιτιολόγηση για το πώς προέκυψε το νέο αυτό δέντρο.

Β. Ας θεωρήσουμε ένα *ετικετοποιημένο* δυαδικό δέντρο T των n κόμβων, δηλαδή ένα δυαδικό δέντρο T στους κόμβους του οποίου έχουν τεθεί οι αριθμοί (*ετικέτες*) από το 1 έως και το n. Εφαρμόζοντας στο δέντρο αυτό τις τρεις διαπεράσεις δυαδικών δέντρων που γνωρίζουμε, δηλαδή την *προδιατεταγμένη*, την *ενδοδιατεταγμένη* και την *μεταδιατεταγμένη*, λαμβάνουμε ένα σύνολο, το *σύνολο διαπεράσεων* Δ_T. Αυτό το σύνολο περιέχει τρεις διαφορετικές διατάξεις των αριθμών που περιέχονται στους κόμβους του δέντρου, μία διάταξη για κάθε διαπέραση.

Προδιατεταγμένη: $\pi_1, \pi_2, \ldots, \pi_n$

Ενδοδιατεταγμένη: $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$

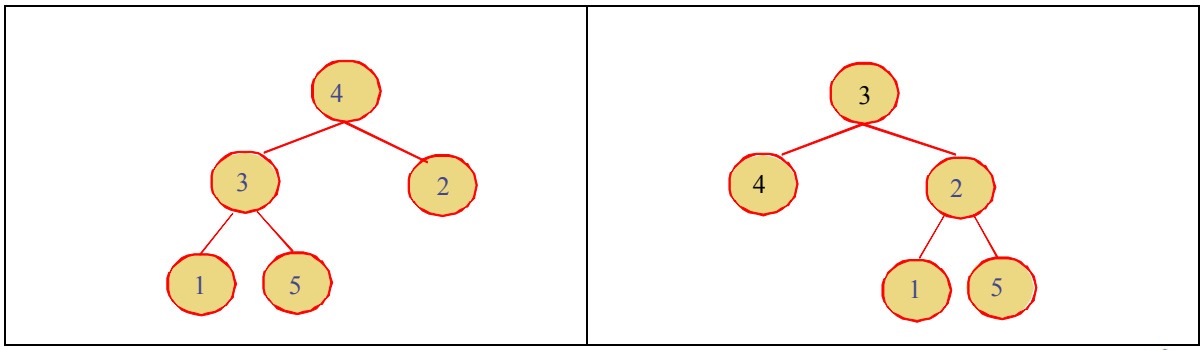
Μεταδιατεταγμένη: $\mu_1, \mu_2, \ldots, \mu_n$

Έστω *δύο* ετικετοποιημένα δυαδικά δέντρα T και T' των n κόμβων το καθένα. Τότε με βάση τα σύνολα διαπεράσεών τους Δ_T και $\Delta_{T'}$ αντίστοιχα, ορίζουμε την εξής συνάρτηση *απόστασης διαπεράσεων* μεταξύ των δύο δέντρων:

$$D(T,T') = \sum_{i=1}^{n}(\pi_i - \pi_i')^2 + \sum_{i=1}^{n}(\varepsilon_i - \varepsilon_i')^2 + \sum_{i=1}^{n}(\mu_i - \mu_i')^2.$$

Με άλλα λόγια, εφαρμόζουμε τις τρεις διαπεράσεις για τα δύο δέντρα χωριστά, και μετά υπολογίζουμε την πιο πάνω αλγεβρική παράσταση για τις *διαφορές* των αντίστοιχων αριθμών που προέκυψαν από τις αντίστοιχες διαπεράσεις των δύο δέντρων.

Για παράδειγμα, έστω τα εξής δύο ετικετοποιημένα δυαδικά δέντρα, T και T', για $n=5$:



Τότε έχουμε τα εξής σύνολα διαπεράσεων, Δ_T και $\Delta_{T'}$, αντίστοιχα, των δέντρων T και T':

Δ_T:

Προδιατεταγμένη: π_1=4, π_2=3, π_3=1, π_4=5, π_5=2

Ενδοδιατεταγμένη: ε_1=1, ε_2=3, ε_3=5, ε_4=4, ε_5=2

Μεταδιατεταγμένη: μ_1=1, μ_2=5, μ_3=3, μ_4=2, μ_5=4

$\Delta_{T\prime}$:

Προδιατεταγμένη: π'_1=3, π'_2=4, π'_3=2, π'_4=1, π'_5=5

Ενδοδιατεταγμένη: ε'_1=4, ε'_2=3, ε'_3=1, ε'_4=2, ε'_5=5

Μεταδιατεταγμένη: μ'_1=4, μ'_2=1, μ'_3=5, μ'_4=2, μ'_5=3

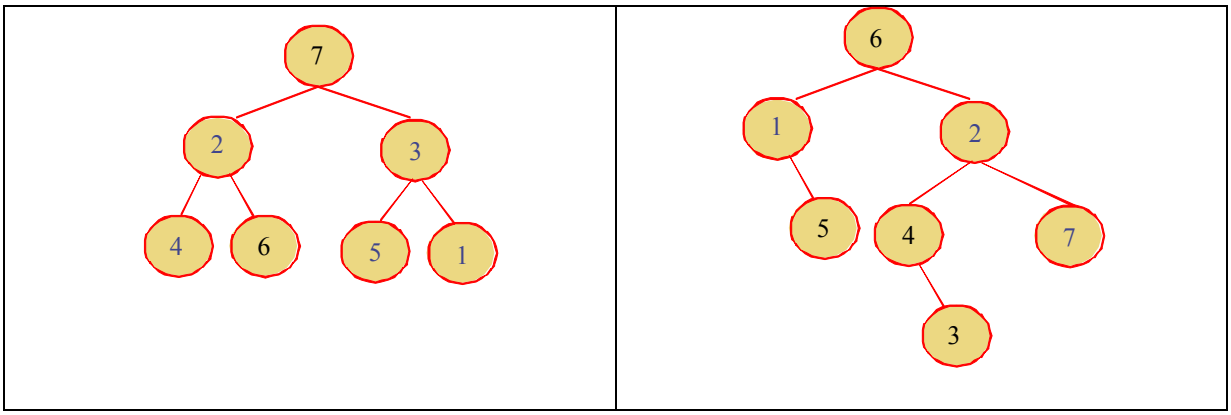
Τότε, με βάση τα πιο πάνω, έχουμε τα εξής (με πράσινο χρώμα φαίνεται το άθροισμα των τετραγώνων των διαφορών των αριθμών από τις προδιατεταγμένες διαπεράσεις των δέντρων, με κόκκινο από τις ενδοδιατεταγμένες και με μπλε από τις μεταδιατεταγμένες):

$$D(T,T') = \sum_{i=1}^{5}(\pi_i - \pi'_i)^2 + \sum_{i=1}^{5}(\varepsilon_i - \varepsilon'_i)^2 + \sum_{i=1}^{5}(\mu_i - \mu'_i)^2 =$$

$$D(T,T') = (4-3)^2 + (3-4)^2 + (1-2)^2 + (5-1)^2 + (2-5)^2 +$$
$$(1-4)^2 + (3-3)^2 + (5-1)^2 + (4-2)^2 + (2-5)^2 +$$
$$(1-4)^2 + (5-1)^2 + (3-5)^2 + (2-2)^2 + (4-3)^2 = 96.$$

Με βάση τα πιο πάνω, απαντήστε στα εξής:

i) Να υπολογίσετε *αναλυτικά* (όπως στο παράδειγμα που δόθηκε προηγουμένως) την *απόσταση διαπεράσεων* $D(T,T')$ για τα εξής δύο δυαδικά δέντρα T και T':



ii) Σε ποιες περιπτώσεις ισχύει $D(T,T') = 0$ για δύο δυαδικά δέντρα n κόμβων; Επιχειρηματολογήστε με συντομία.

ΥΠΟΕΡΓΑΣΙΑ 4 (βαθμοί 25)

Σκοπός της άσκησης αυτής είναι να μελετηθεί πως η γλώσσα C μπορεί να χρησιμοποιηθεί για να υλοποιήσει μηχανισμούς εκχώρησης *μνήμης* σε προγράμματα τα οποία εκτελούνται.

Ο βασικός τρόπος παρακολούθησης της μνήμης είναι η διατήρηση μίας *απλά συνδεδεμένης* λίστας με τα *ελεύθερα* τμήματα της μνήμης όπου ο κάθε κόμβος της περιέχει την *αρχική διεύθυνση* του τμήματος, *το μέγεθος του σε bytes* και *δείκτη* προς τον κόμβο που αναπαριστά το επόμενο *ελεύθερο* τμήμα. Η συγκεκριμένη λίστα είναι διατεταγμένη ως προς τις αρχικές διευθύνσεις των τμημάτων μνήμης και αρχικά αποτελείται από ένα μόνο κόμβο με αρχική διεύθυνση τμήματος 0 και με μέγεθος τμήματος το μέγεθος της μνήμης. Οι βασικές λειτουργίες που ο μηχανισμός αυτός παρέχει είναι: (i) ***δέσμευση*** ενός τμήματος μνήμης για ένα πρόγραμμα προς εκτέλεση και αφαίρεσή του από τη συνδεδεμένη λίστα και (ii) ***αποδέσμευση*** του τμήματος μνήμης που μία διεργασία κατέχει και επιστροφή του στην απλά συνδεδεμένη λίστα. Η επιστροφή ενός τμήματος στη λίστα ελεύθερων τμημάτων πραγματοποιείται με τοποθέτηση του νέου τμήματος μέσα στη λίστα ώστε τα τμήματα να είναι διατεταγμένα με βάση την αρχική διεύθυνση τους και στη συνέχεια έλεγχος αν το δοθέν τμήμα μπορεί να συγχωνευθεί με ένα ή και τα δύο γειτονικά του τμήματα για να δημιουργηθεί ένα μεγαλύτερο τμήμα. Αν ναι, τότε γίνεται συγχώνευση.

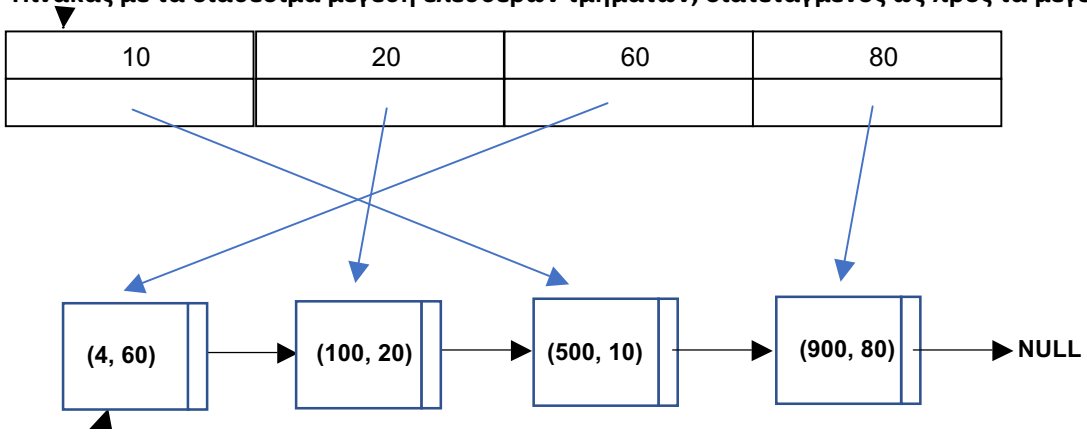
Για τη δέσμευση μνήμης συγκεκριμένου μεγέθους έχουν προταθεί πολλοί διαφορετικοί αλγόριθμοι. Στη συγκεκριμένη άσκηση θα εστιάσουμε στον **αλγόριθμο βέλτιστου ταιριάσματος** ο οποίος αναζητεί ολόκληρη τη λίστα *μέχρι να βρει το μικρότερο τμήμα από όλα της λίστας στο οποίο να χωράει η απαίτηση μνήμης της διεργασίας*. Αν αυτό ισχύει για περισσότερα από ένα τμήματα (ίδιο μέγεθος) επιλέγεται οποιοδήποτε από αυτά. Στη συνέχεια δεσμεύει το χώρο μνήμης που της αντιστοιχεί *ξεκινώντας από την αρχική διεύθυνση* του τμήματος, και *αφήνει* στη λίστα το υπολειπόμενο κομμάτι μνήμης. Πιο αναλυτικά, εφόσον ο κόμβος με το κατάλληλο ελεύθερο τμήμα μνήμης βρεθεί, αν το μέγεθός του είναι *ίσο* με το μέγεθος που ζητείται τότε ο κόμβος *αφαιρείται* από τη λίστα, διαφορετικά ο κόμβος με το ελεύθερο τμήμα δεν αφαιρείται αλλά παραμένει αναπαριστώντας το υπολειπόμενο κομμάτι μνήμης.

Στις συνήθεις υλοποιήσεις η εύρεση του καλύτερου τμήματος πραγματοποιείται με σειριακή αναζήτηση στη λίστα ελεύθερων τμημάτων μνήμης. Στη συγκεκριμένη υλοποίηση θα θεωρήσουμε ότι χρησιμοποιούμε ***ένα βοηθητικό πίνακα*** που αποθηκεύει ***διατεταγμένα*** τα ελεύθερα τμήματα μνήμης ***με βάση το μέγεθός τους σε bytes*** οπότε με δυαδική αναζήτηση με βάση το αιτούμενο μέγεθος εντοπίζεται το κατάλληλο τμήμα μνήμης. Ένθεση και διαγραφή στον πίνακα, ώστε να παραμένει διατεταγμένος, γίνεται με εύρεση της κατάλληλης θέσης εισαγωγής ή διαγραφής και μετά μετακίνηση των στοιχείων του πίνακα προς τα δεξιά (για να ελευθερωθεί η θέση για το προς εισαγωγή στοιχείο) ή προς τα αριστερά (για να συμπληρωθεί η κενή θέση που δημιουργείται από την διαγραφή στοιχείου). Υπάρχουν πιο αποδοτικοί

τρόποι ενημέρωσης αλλά εδώ μας ενδιαφέρει η υλοποίηση της βασικής ιδέας. Στην πιο απλή περίπτωση ο πίνακας αποτελείται από στοιχεία με δύο πεδία: μία ακέραια τιμή που υποδηλώνει το μέγεθος του τμήματος μνήμης σε bytes και ένα δείκτη στον ***αντίστοιχο*** κόμβο της συνδεδεμένης λίστας με τα *ελεύθερα* τμήματα. Ο πίνακας και η λίστα θα πρέπει *να είναι συγχρονισμένα*, ώστε όταν ενημερώνεται η μία δομή να ενημερώνεται και η άλλη.

Στο κάτωθι σχήμα, παρουσιάζεται η σύνδεση των προαναφερθεισών δομών δεδομένων:

Πίνακας με τα διαθέσιμα μεγέθη ελεύθερων τμημάτων, διατεταγμένος ως προς τα μεγέθη:



Λίστα ελεύθερων τμημάτων μνήμης, διατεταγμένη ως προς την αρχική διεύθυνση μνήμης κάθε τμήματος (σε κάθε κόμβο το ζευγάρι σημαίνει (αρχική διεύθυνση, μέγεθος τμήματος)).

Θεωρείται ότι τόσο η λίστα με τα ελεύθερα τμήματα μνήμης όσο και ο πίνακας δηλώνονται ως *καθολικές μεταβλητές* ορατές από κάθε συνάρτηση. Θεωρήστε ότι το αρχικό μέγεθος της ελεύθερης μνήμης είναι 1000 bytes (από αρχική διεύθυνση 0) και δηλώνεται σαν μία συμβολική σταθερά με όνομα *N*. Για τον βοηθητικό πίνακα θεωρήστε ότι το μέγιστο μέγεθός του είναι *N/2+1*, και ότι το τρέχον πλήθος στοιχείων του δίνεται από μία καθολική μεταβλητή με όνομα `free_items`.

Με βάση τις ανωτέρω υποθέσεις ζητούνται τα ακόλουθα:

Α. Να υλοποιηθεί μία συνάρτηση με όνομα `void printfreelist()` που να εκτυπώνει το περιεχόμενο της λίστας των ελεύθερων τμημάτων μνήμης, διαπερνώντας την από την κεφαλή προς το τέλος της.

Β. Να υλοποιηθεί μία συνάρτηση δέσμευσης μνήμης, με όνομα `int bestfit(int alloc)` η οποία να δέχεται ως όρισμα ένα ***ακέραιο*** αριθμό που θα είναι η απαίτηση σε bytes μνήμης ενός προγράμματος και να επιστρέφει την αρχική διεύθυνση του τμήματος μνήμης που είναι διαθέσιμο με κατάλληλη εφαρμογή του **αλγορίθμου βέλτιστου ταιριάσματος.** Στη συνέχεια η συνάρτηση να δεσμεύει το ελεύθερο τμήμα. Αν δεν υπάρχει τμήμα ελεύθερης μνήμης που να ικανοποιεί το αντίστοιχο μέγεθος να επιστρέφει -1. Η εύρεση του τμήματος γίνεται με χρήση (δυαδική αναζήτηση) του πίνακα και στη συνέχεια ενημέρωση λίστας και πίνακα. Πιο

συγκεκριμένα αφού το κατάλληλο ελεύθερο τμήμα μνήμης βρεθεί, αν το μέγεθός του είναι ακριβώς ίσο με το μέγεθος που ζητείται τότε ο κόμβος ελεύθερης μνήμης αφαιρείται από τη λίστα *και ενημερώνεται ο πίνακας διαγράφοντας το αντίστοιχο στοιχείο,* διαφορετικά ο κόμβος με το ελεύθερο τμήμα δεν αφαιρείται αλλά παραμένει αναπαριστώντας το υπολειπόμενο κομμάτι μνήμης και *ενημερώνεται και το αντίστοιχο στοιχείο του πίνακα για το νέο μέγεθος της μνήμης που αναπαριστά το τμήμα.*

Γ. Να υλοποιηθεί μία συνάρτηση αποδέσμευσης μνήμης, με όνομα `void returntofreelist(int address, int size)` που να επιστρέφει στην διαθέσιμη μνήμη ένα τμήμα μνήμης με αρχική διεύθυνση *address* και μέγεθος *size bytes*. Η τοποθέτηση του νέου τμήματος μέσα στη λίστα γίνεται *με βάση τη διεύθυνσή του*, ώστε η λίστα να παραμένει διατεταγμένη με βάση τις διευθύνσεις και πραγματοποιείται διατρέχοντας την από την κεφαλή της μέχρι να βρεθεί η θέση ένθεσης. *Πριν την ένθεση του τμήματος γίνεται έλεγχος αν μπορεί να συγχωνευθεί με τα δύο γειτονικά του τμήματα για να δημιουργηθεί ένα μεγαλύτερο τμήμα.* Αν το νέο τμήμα μπορεί να συγχωνευθεί *μόνο* με το προηγούμενο ή *μόνο* με το επόμενο τμήμα τότε εκτός από τους κόμβους της λίστας πρέπει να ενημερωθούν και τα αντίστοιχα στοιχεία του πίνακα με το νέο μέγεθος τους. Προσέξτε ότι αν η συγχώνευση γίνει με το επόμενο τμήμα τότε αλλάζει και η αρχική διεύθυνση στον αντίστοιχο κόμβο της λίστας. Αν το νέο τμήμα μαζί *και με τα δύο* γειτονικά ενωθούν σε ΕΝΑ ενιαίο τμήμα τότε το επόμενο τμήμα πρέπει να διαγραφεί από τον πίνακα και από τη λίστα και το προηγούμενο τμήμα να ενημερωθεί με το συνολικό νέο μέγεθος σε πίνακα και λίστα. *Αν δεν γίνει συγχώνευση* τότε μετά την ένθεση του νέου τμήματος στη λίστα πρέπει να γίνει και ένθεση ενός νέου στοιχείου στον πίνακα με δείκτη στο νέο τμήμα.

Προσέξτε ότι στην περιγραφή των συναρτήσεων τα στοιχεία του πίνακα μερικές φορές *αλλάζουν* μέγεθος. Επειδή ο πίνακας πρέπει να είναι διατεταγμένος θα πρέπει μετά από κάθε αλλαγή μεγέθους στοιχείου και αν η διάταξη επηρεάζεται, είτε να γίνεται διάταξη όλου του πίνακα, είτε να πραγματοποιείται *διαγραφή και στη συνέχεια εκ νέου ένθεση* του στοιχείου στον πίνακα με το νέο μέγεθός του. Στο σχέδιο προγράμματος με τίτλο `ypoergasia4_code_template.c` καλείστε να συμπληρώσετε τον κώδικα των συναρτήσεων που περιγράφηκαν, *ενώ μπορείτε να ορίσετε και δικές σας βοηθητικές συναρτήσεις.*

<u>Γενικές Υποδείξεις</u>:

Ι) Για τις απαντήσεις της εργασίας μπορείτε να ανατρέξετε στην συμπληρωματική βιβλιογραφία που δίνεται και στα βοηθητικά κείμενα που υπάρχουν στον δικτυακό τόπο /

portal της θεματικής ενότητας. Συνιστάται να προσθέσετε στο τέλος της εργασίας σας κατάλογο βιβλιογραφίας.

II) Τρόπος παράδοσης εργασίας:

Οι απαντήσεις πρέπει να είναι γραμμένες με χρήση επεξεργαστή κειμένου (π.χ. Word) σε σελίδες διαστάσεων Α4. Το αρχείο να περιέχει ως πρώτη σελίδα το κείμενο του Εντύπου Υποβολής - Αξιολόγησης και ως δεύτερη σελίδα τον τίτλο «Σχόλια προς τον φοιτητή» (θα συμπληρωθεί από τον καθηγητή σας). Οι απαντήσεις στις υπο-εργασίες θα αρχίζουν από την τρίτη σελίδα, χωρίς να επαναλαμβάνονται οι εκφωνήσεις. Κάθε υπο-εργασία θα αρχίζει από νέα σελίδα. Για την απάντησή σας θα πρέπει να χρησιμοποιείτε υποχρεωτικά το **Πρότυπο Υποβολής Γραπτής Εργασίας**. Ενδεικτικά, οι απαντήσεις μπορούν να επιτευχθούν σε περίπου 17 σελίδες.

III) Η καλή παρουσίαση της εργασίας λαμβάνεται υπόψη στην αξιολόγηση της εργασίας.

IV) Αμυντικός προγραμματισμός και σχολιασμός κώδικα:

Ο κώδικας πρέπει να είναι επαρκώς σχολιασμένος και να χρησιμοποιεί στοιχεία αμυντικού προγραμματισμού, όπου αυτό ζητείται. Για την εφαρμογή του αμυντικού προγραμματισμού αρκεί να γίνεται έλεγχος ως προς το εάν μία τιμή που εισάγεται από το χρήστη του προγράμματος βρίσκεται εντός αποδεκτών ορίων (π.χ. να είναι θετική, διαφορετική από το 0, μεγαλύτερη από 20 κ.λπ.). Δεν απαιτείται έλεγχος ως προς το εάν η τιμή ανήκει στο σωστό τύπο δεδομένων (π.χ. ακέραιος, πραγματικός αριθμός, χαρακτήρας κ.λπ.) σύμφωνα με τον τύπο της μεταβλητής στην οποία θα αποθηκευτεί η τιμή αυτή.

V) Τρόπος παράδοσης κώδικα:

Στα ερωτήματα που ζητείται υλοποίηση κώδικα στη γλώσσα προγραμματισμού C (ANSI C), για να θεωρηθούν οι απαντήσεις σας ολοκληρωμένες θα πρέπει:

- Ο κώδικας (όπου ζητείται) να είναι επαρκώς σχολιασμένος και ενσωματωμένος μέσα στο .doc αρχείο του Word με τις απαντήσεις σας καθώς *και* σε ξεχωριστό .c αρχείο κειμένου, όχι Word.
- Το όνομα κάθε .c αρχείου να περιλαμβάνει το επώνυμό σας με λατινικούς χαρακτήρες, το χαρακτήρα της υπογράμμισης και τον αριθμό του συγκεκριμένου υποερωτήματος (π.χ. αν το επώνυμό σας είναι Γεωργίου, τότε ο κώδικας για την υποεργασία 1β θα έχει το όνομα Georgiou_1b.c).
- Κάθε αρχείο C που θα παραδοθεί θα πρέπει τουλάχιστον να περνάει τη φάση της μεταγλώττισης χωρίς λάθη.
- Όλα τα .c αρχεία με τον πηγαίο κώδικα και το .doc αρχείο κειμένου να υποβληθούν στη διεύθυνση **http://study.eap.gr**.