



Μπορείτε να έχετε μία κόλλα σημειώσεων **A4** με ευδιάκριτο το ονοματεπώνυμό σας. Τα θέματα επιστρέφονται, μαζί με το γραπτό, την κόλλα σημειώσεών σας, και όσα πρόχειρα χρειαστήκατε!

Ονοματεπώνυμο

A.M.

Θέμα 1 [8 + 12 = 20]

α) Εξηγείστε με απλά λόγια τι κάνει η διπλανή αναδρομική συνάρτηση. Προσοχή, δεν σας ζητείται να περιγράψετε τον κώδικα, αλλά πείτε τη λειτουργία που επιτελεί (π.χ., ελέγχει αν ένα αλφαριθμητικό είναι καρκινικό).

β) Να γραφτεί αναδρομική συνάρτηση `stars` η οποία παίρνει ως είσοδο έναν θετικό ακέραιο `N` και τυπώνει στην πρώτη γραμμή ένα *, στη δεύτερη δύο **, κ.ο.κ. μέχρι τη `N`-οστή γραμμή που θα έχει `N *`. Έτσι αν για παράδειγμα η `stars` κληθεί με όρισμα 5 θα πρέπει να τυπώσει το διπλανό pattern.

```
void mystery(char *str, int k) {  
    char tmp;  
  
    if (k < 2) return;  
    k = k - 1;  
    tmp = *str;  
    *str = str[k];  
    str[k] = tmp;  
  
    mystery(str + 1, k - 1);  
}
```

Θέμα 2 [10 + 15 = 25]

Δίνεται η διπλανή δομή η οποία χρησιμοποιείται για να αποθηκεύσει `market pass` πολιτών σε διαφόρων ειδών λίστες Συμπληρώστε κατάλληλα τη δομή και δώστε την υλοποίηση της συνάρτησης διαγραφής remove() για κάθε μία από τις παρακάτω περιπτώσεις.

```
struct mPass {  
    int passID;  
    float balance;  
    ...  
};
```

α) Η συνάρτηση `struct mPass *remove(struct mPass *head)` διαγράφει το `marketPass` που βρίσκεται στο τέλος μία διπλά συνδεδεμένη κυκλικής λίστας.

β) Η συνάρτηση `struct mPass *remove(struct mPass *head, int bal)` διαγράφει όλα τα `marketPass` με υπόλοιπο `balance` μεγαλύτερο από `bal` (δεύτερο όρισμα της `remove`) από μία απλά συνδεδεμένη κυκλική λίστα.

Και στα δύο ερωτήματα να θεωρήσετε ότι το πρώτο όρισμα της συνάρτησης `remove` είναι η κεφαλή της λίστας και δεν έχετε άλλους δείκτες σε άλλα στοιχεία της λίστας (π.χ., `tail`). Προσέξτε οι συναρτήσεις διαγραφής που θα γράψετε να απελευθερώνουν μνήμη, να φτιάχνουν όλους τους δείκτες που αλλάζουν, και να επιστρέφουν τη (νέα) κεφαλή της λίστας.

Θέμα 3 [5 + 8 + 12 = 25]

Έστω η δομή `struct pBook{ ... }`; η οποία χρησιμοποιείται ως τηλεφωνικός κατάλογος και αποθηκεύει ένα ονοματεπώνυμο, ένα τηλέφωνο και ημερομηνία γέννησης. Ειδικά για την ημερομηνία γέννησης θα πρέπει να αποθηκεύουμε χωριστά την ημέρα, το μήνα και το έτος ως `int` σε μία άλλη δομή `struct date{ ... }`; η οποία στη συνέχεια θα χρησιμοποιείται εντός της `struct pBook{ ... }`;

α) Γράψτε τις δύο δομές και συμπληρώστε κατάλληλα τα πεδία που θα πρέπει να έχουν ώστε να καλύπτουν τις παραπάνω απαιτήσεις.

β) Αποθηκεύστε ότι δεδομένα θέλετε για εσάς ή έναν φίλο σας στη μεταβλητή `struct pBook person` χρησιμοποιώντας τον τελεστή `"."` και κατόπιν κάντε το ίδιο χρησιμοποιώντας τον τελεστή `"->"`.

γ) Αποθηκεύστε ότι δεδομένα θέλετε για έναν φίλο σας στη μεταβλητή `struct pBook *person` χρησιμοποιώντας τον τελεστή `"."` και κατόπιν κάντε το ίδιο χρησιμοποιώντας τον τελεστή `"->"`. Στην περίπτωση αυτή προσέξτε ότι η μεταβλητή `person` είναι δείκτης!

Θέμα 4 [15]

Να γραφεί πρόγραμμα το οποίο θα ζητάει μια φράση (μέχρι 1000 χαρακτήρες) από τον χρήστη, θα τη διαβάζει από το πληκτρολόγιο, θα τη χωρίζει σε λέξεις, θα επεξεργάζεται κάθε λέξη ως εξής: αν η λέξη έχει λιγότερα από δύο γράμματα θα την αγνοεί, θα κάνει το πρώτο γράμμα της λέξης κεφαλαίο και όλα τα υπόλοιπα πεζά και θα εμφανίζει τις επιμέρους αλλαγμένες λέξεις στην οθόνη. Οτιδήποτε στη φράση δεν είναι γράμμα θα πρέπει να αγνοείται.

Για παράδειγμα, αν ο χρήστης πληκτρολογήσει: "This is the BEST exam...!"
η έξοδος του προγράμματος θα πρέπει να είναι: This The Best Exam
με τις λέξεις τυπωμένες μία ανά γραμμή. Η επεξεργασία θα πρέπει να γίνει με τη συνάρτηση `strtok()`, απαντήσεις χωρίς αυτή δε λαμβάνουν βαθμούς.

Θέμα 5 [15 + 10 = 25]

α) Να γραφτεί πρόγραμμα, το οποίο θα ζητάει και θα διαβάζει ως είσοδο από το πληκτρολόγιο δύο ονόματα (δυσδικών) αρχείων. Στη συνέχεια, θα αντιγράφει από το πρώτο αρχείο (αρχείο πηγής) στο δεύτερο (αρχείο προορισμού) έναν τυχαίο αριθμό από bytes μεταξύ του 3 και του 45 (που θα παράγεται με κατάλληλο τρόπο), και αυτή η διαδικασία θα συνεχίζεται έως ότου όλο το αρχείο πηγής αντιγραφεί στο αρχείο προορισμού. Προφανώς αν ο αριθμός των byte του αρχείου που είναι να αντιγραφούν ξεπεραστεί, η τελευταία αντιγραφή που θα γίνει θα έχει λιγότερα bytes (όσα υπολείπονται)

Έστω ως παράδειγμα αν ένα αρχείο πηγής είναι 12 bytes, έστω ότι στην πρώτη αντιγραφή ο τυχαίος αριθμός που παράγεται να αντιγραφούν 4 bytes, στη δεύτερη 3, και στην τρίτη 44 οπότε και θα ολοκληρωθεί η διαδικασία αντιγράφοντας τα εναπομείναντα 5 bytes από το αρχείο πηγής σε αυτό του προορισμού. Μην ξεχάσετε τους απαραίτητους ελέγχους κατά το άνοιγμα των αρχείων!

β) Να γραφτεί πρόγραμμα το οποίο θα διαβάζει ως όρισμα γραμμής εντολών έναν ακέραιο N , και θα τυπώνει κατάλληλο μήνυμα λάθους αν μπουν περισσότερα ή λιγότερα ορίσματα. Στη συνέχεια θα δημιουργεί έναν τετραγωνικό πίνακα $N \times N$ όπως φαίνεται δίπλα.

0	1	1	1	1
-1	0	1	1	1
-1	-1	0	1	1
-1	-1	-1	0	1
-1	-1	-1	-1	0