

1. Να ορίσετε μία δομή που θα αποθηκεύει τα στοιχεία ενός τραπεζικού λογαριασμού με πεδία: όνομα καταθέτη, ποσό κατάθεσης και ημερομηνία κατάθεσης. Γράψτε πρόγραμμα το οποίο:
  - (i) θα δηλώνει ένα πίνακα με το όνομα `accounts` που θα αποθηκεύει ένα σύνολο η τραπεζικών λογαριασμών,
  - (ii) θα διαβάζει τα στοιχεία των η λογαριασμών από τον χρήστη,
  - (iii) θα διαβάζει το όνομα ενός καταθέτη και θα εκτυπώνει όλες τις πληροφορίες για όλους τους λογαριασμούς με το συγκεκριμένο όνομα. Αν οι καταθέσεις είναι περισσότερες από μία θα εκτυπώνει επίσης το άθροισμα όλων των ποσών κατάθεσης.

Τα ζητούμενα (ii) και (iii) να γίνουν με χρήση κατάλληλων συναρτήσεων και η υλοποίηση του πίνακα προτείνεται να γίνει με δείκτες (αν και δεν είναι απαραίτητο).

2. Μία εταιρεία θέλει να καταχωρεί τα στοιχεία των υπαλλήλων της (`N` σε σύνολο), και στην συνέχεια να εμφανίζεται στην οθόνη το ονοματεπώνυμο των υπαλλήλων που έχουν το μεγαλύτερο μισθό και ποιος είναι αυτός ο μισθός, το ονοματεπώνυμο των υπαλλήλων που έχουν το μεγαλύτερο σύνολο πωλήσεων και ποιο είναι αυτό το σύνολο καθώς και το ονοματεπώνυμο των υπαλλήλων που έχουν το μεγαλύτερο έτος πρόσληψης και ποιο έτος είναι αυτό. Υλοποιήστε κώδικα σε `C` που πραγματοποιεί τις ανωτέρω λειτουργίες και όπου τα στοιχεία των υπαλλήλων αποθηκεύονται σε μία απλή λίστα (υλοποιημένη ως πίνακας).

Στο πρόγραμμα να χρησιμοποιείται δομή με τα παρακάτω στοιχεία υπαλλήλων (υποθέτουμε ότι κάθε υπάλληλος απασχολείται μόνο σε ένα τμήμα:

Όνομα υπαλλήλου

Επώνυμο υπαλλήλου

Μισθός υπαλλήλου

Σύνολο πωλήσεων

Έτος πρόσληψης

3. Μία από τις πιο σημαντικές χρήσεις της γλώσσας `C` είναι στη διαχείριση αλφαριθμητικών, και αυτό γιατί η `C` παρέχει πολλές ευκολίες στη διαχείρισή τους. Σκοπός της συγκεκριμένης άσκησης είναι η διερεύνηση αυτών των δυνατοτήτων, για τον αποτελεσματικό χειρισμό ενός πίνακα αλφαριθμητικών. Θεωρήστε λοιπόν ένα πίνακα `N` θέσεων όπου σε κάθε θέση αποθηκεύεται η τιμή μίας μεταβλητής τύπου δομής με δύο μέλη ένα αλφαριθμητικό και μία ακέραια τιμή που υποδηλώνει το μήκος του αλφαριθμητικού. Θεωρείστε ότι κάθε αλφαριθμητικό

είναι το καθένα μέγιστου μήκους M-1 και αποτελείται μόνο από αριθμητικούς και αλφαριθμητικούς χαρακτήρες (όλοι του Αγγλικού αλφαβήτου), χωρίς κενά, με το N και το M να δηλώνονται στο πρόγραμμα σαν συμβολικές σταθερές. Έστω `word_table` το όνομα του πίνακα.

Τυπικά η δήλωση θα δίνεται ως εξής:

```
struct word_pair
{
    char word[M]; int length;
};
struct word_pair word_table[N]
```

Ζητείται να υλοποιηθούν οι εξής συναρτήσεις:

A. Μία συνάρτηση που θα ζητάει από τον χρήστη και θα γεμίζει τον πίνακα με N αλφαριθμητικά, και για κάθε αλφαριθμητικό θα υπολογίζει και θα αποθηκεύει στην αντίστοιχη θέση το μήκος του. Το μήκος του αλφαριθμητικού θα πρέπει να υπολογίζεται με δική σας συνάρτηση και όχι με συνάρτηση της βιβλιοθήκης της C. Η συνάρτηση να δηλωθεί ως *void initialize(struct word\_pair word\_table[], int size)* (με size συμβολίζεται ο αριθμός των γραμμών του πίνακα). Η είσοδος των συμβολοσειρών να γίνει με αμυντικό προγραμματισμό.

B. Μία συνάρτηση που θα δέχεται σαν είσοδο δύο αλφαριθμητικά, και θα αντικαθιστά κάθε εμφάνιση του πρώτου αλφαριθμητικού στα αλφαριθμητικά του πρώτου πίνακα με το δεύτερο αλφαριθμητικό. Η συνάρτηση να δηλωθεί ως :

*void replace(struct word\_pair word\_table[], int size, char search\_string[M], char replacement\_string[M])* (με size συμβολίζεται ο αριθμός των γραμμών του πίνακα, `search_string` το αλφαριθμητικό που αναζητούμε, και `replacement_string` το αλφαριθμητικό με το οποίο γίνεται η αντικατάσταση) .

Γ. Μία συνάρτηση που δοθείσης ενός αλφαριθμητικού εισόδου θα εντοπίζει και θα εκτυπώνει όλες τις γραμμές του πίνακα που την περιέχουν σαν υποσυμβολοσειρά (ο όρος υπονοεί συνεχόμενες εμφανίσεις των συμβόλων) και σαν υποακολουθία (ο όρος δεν απαιτεί συνεχόμενες εμφανίσεις). Η συνάρτηση να δηλωθεί σαν *void sub\_string\_sequence(struct word\_pair word\_table[], char search\_string[M])* (με size συμβολίζεται ο αριθμός των γραμμών του πίνακα, με `search_string` το αλφαριθμητικό που αναζητούμε)

Δ. Χρησιμοποιώντας τον παρακάτω αρχικοποιημένο λεξικό αλφαριθμητικών *lexicon* για μετάφραση μεταξύ Ιταλικών και Αγγλικών λέξεων υλοποιήστε μία συνάρτηση που μεταφράζει κάθε αλφαριθμητικό του `word_table` στο αντίστοιχο ιταλικό αν υπάρχει στο λεξικό, διαφορετικά το αφήνει ως έχει. Η

συνάρτηση να δηλωθεί ως: *void translate(struct word\_table word\_table[], int size)* (με *size* συμβολίζεται ο αριθμός των γραμμών του πίνακα)

```
char lexicon[][2][M]={
{"table", "tavolo"},
{"word", "parola"},
{"car", "auto"},
{"tree", "albero"},
{"addition", "aggiuntaguidare"},
{"drive", "fahern"},
{"bicycle", "macchina"}
}
```

Ελέγξτε τον κώδικά σας με ένα κυρίως πρόγραμμα το οποίο θα καλεί τις ανωτέρω συναρτήσεις με την σειρά με την οποία ορίστηκαν.

#### **Υπόδειγμα:**

Προτείνεται η χρήση των συναρτήσεων:

- *int isalnum(int ch)* της βιβλιοθήκης <ctype.h> που επιστρέφει μη μηδενική τιμή (αληθή) αν ο χαρακτήρας *ch* είναι αριθμητικό ψηφίο ή γράμμα (πεζό ή κεφαλαίο) του αγγλικού αλφαβήτου, διαφορετικά επιστρέφει 0
- *char \*strcpy(char \*s, const char \*t)*, συνάρτηση της βιβλιοθήκης <string.h> που αντιγράφει το αλφαριθμητικό *t* στο αλφαριθμητικό *s*.