Μεταγλώττιση: Ο κώδικας θα πρέπει να μεταγλωττίζεται χωρίς **warnings**. Χαρακτηρισμός “μη ικανοποιητικό” σημαίνει ότι είχατε τουλάχιστον ένα warning. Κώδικας που παράγει λάθη (errors) κατά τη μεταγλώττιση βαθμολογείται με FAIL.

Στοίχιση: Ο κώδικας πρέπει να είναι στοιχισμένος σωστά, με ιδιαίτερη προσοχή στη στοίχιση εμφωλευμένων for/while/if. Χαρακτηρισμός “μέτρια” γενικά σημαίνει ότι έχετε κατά το πλείστον καλή στοίχιση Χαρακτηρισμός “μη ικανοποιητική” σημαίνει ότι δεν έχετε στοιχίσει τον κώδικά σας ή η στοίχιση είναι πολύ ασυνεπής ειδικά όσον αφορά τις δομές ελέγχου/επανάληψης.

Ονόματα: Όπως πάντα, θέλουμε περιγραφικά ονόματα συναρτήσεων και μεταβλητών. Για παράδειγμα, το όνομα function ή sinartisi για μία συνάρτηση δεν είναι καλό γιατί δε μας λέει τι κάνει η συγκεκριμένη συνάρτηση. Ανάλογα τα ονόματα a, b, m, x, y, z, variable, metabliti, array κλπ δεν είναι καλά ονόματα μεταβλητών διότι δεν προσδιορίζουν το σκοπό για τον οποίο χρησιμοποιείται η συγκεκριμένη μεταβλητή.

- Συνιστάται να διαλέγουμε ρηματικές φράσεις ως ονόματα συναρτήσεων διότι κάθε συνάρτηση κάνει κάποια ενέργεια. Για παράδειγμα get_limit και όχι απλά limit. Αντίστοιχα συνιστάται να διαλέγουμε ουσιαστικά ως ονόματα μεταβλητών, όπως limit (όριο), sum (άθροισμα) κλπ.
- Ονόματα τοπικών μεταβλητών διαφορετικών συναρτήσεων μπορούν να είναι ίδια, και ενδείκνυται κάτι τέτοιο αν πρόκειται για όμοιες ποσότητες. Ονόματα του ενός γράμματος πρέπει να αποφεύγονται με μόνη εξαίρεση μετρητές για for ή while loops.

Συνάρτηση εισαγωγής των στοιχείων ενός φοιτητή στον πίνακα

Το **prototype** της συνάρτησης διαμορφώνεται ως εξής:

```
void append_student(struct student students[], unsigned int *size)
```

Υλοποίηση

Βήμα 1ο - Έλεγχος εάν ο αριθμός των εγγραφών είναι ίσος με το μέγεθος του πίνακα:

```
if(*size==MAX_STUDENTS) {
      printf("No space!\n");
      return;
}
```

Βήμα 2ο - Ανάγνωση με χρήση sprintf: Στην άσκηση έχετε να διαβάσετε εκτός της συμβολοσειράς έναν μη προσημασμένο ακέραιο και 3 αριθμούς κινητής υποδιαστολής. Η θέση του πίνακα στην οποία θα διαβάσετε

είναι η επόμενη κενή θέση, δηλαδή η θέση **`*size`**. Μετά την ανάγνωση αρκεί να αυξήσετε το μέγεθος του πίνακα κατά 1 ως εξής **`(*size)++`** ή **`(*size) = (*size)+1;`**

Προσοχή: Το *size++ δεν είναι το ίδιο με το (*size)++ γιατί ο τελεστής ++ έχει μεγαλύτερη προτεραιότητα από τον τελεστή *

Ανάγνωση: Μπορείτε να το κάνετε είτε σε 2 βήματα (πρώτα στο string και μετά όλα τα υπόλοιπα) ή όλα σε ένα βήμα.

Στην 1η περίπτωση η διαδικασία διαβάσματος έχει ως εξής

```
char format_str[13];
sprintf(format_str, "%%%ds", MAX_NAME_SIZE-1);
scanf(format_str, students[*size].name);
scanf("%u %lf %lf %lf\n",
            &students[*size].aem, &students[*size].quiz[0],
            &students[*size].quiz[1], &students[*size].final);
```

Στην 2η περίπτωση θα χρειαστείτε ένα μεγαλύτερο buffer. Η διαδικασία διαβάσματος είναι ανάλογη με εξαίρεση την sprintf που διαμορφώνεται ως εξής:

```
char format_str[28];
sprintf(format_str, "%%%ds %%u %%lf %%lf %%lf", MAX_NAME_SIZE-1);
```

Συνάρτηση διαγραφής στοιχείων

Το **prototype** της συνάρτησης διαμορφώνεται ως εξής:

```
int delete(struct student students[], unsigned int *size, char *targetname)
```

Υλοποίηση

Η συνάρτηση ελέγχει εάν υπάρχουν εγγραφές με πεδίο **`name`** ίσο με **`targetname`** μέσα στο πίνακα. Εάν τις βρει τις διαγράφει αντιγράφοντας το τελευταίο στοιχείο του πίνακα στην θέση που θέλει να διαγράψει και μειώνει το μέγεθος κατά 1. Για κάθε θέση i του πίνακα κάνουμε τα εξής:

```
if( strcmp(students[i].name, targetname) == 0) {
      students[i] = students[(*size)-1];
      (*size)--; // μείωση των εγγραφών του πίνακα κατά 1.
}
```

Ξεκινάμε την αναζήτηση εγγραφών προς διαγραφή από το τέλος προς την αρχή. Με αυτό τον τρόπο εάν μία ή περισσότερες εγγραφές που βρίσκονται στο τέλος του πίνακα πρέπει να διαγραφούν, αυτές διαγράφονται κατά προτεραιότητα. Για παράδειγμα εάν έχουμε τις εγγραφές: **Giorgos_Thanos, Vana_Doufexi, Christos_Antonopoulos, Giorgos_Thanos** και θελήσουμε να διαγράψουμε την εγγραφή **Giorgos Thanos** τότε εάν ξεκινήσουμε την διαγραφή από την αρχή και χωρίς να παρέμβουμε στο μετρητή i του loop θα έχουμε το εξής αποτέλεσμα: **Giorgos_Thanos, Vana_Doufexi, Christos_Antonopoulos** ενώ εάν ξεκινήσουμε από το τέλος θα λάβουμε το σωστό αποτέλεσμα **Vana_Doufexi, Christos_Antonopoulos**

Η συνάρτηση πρέπει να επιστρέφει 1 εάν έγινε έστω και μία διαγραφή, διαφορετικά μηδέν.

Συνάρτηση αναζήτησης της 1ης εγγραφής στον πίνακα

Το **prototype** της συνάρτησης διαμορφώνεται ως εξής:

```
struct student *search_student(struct student students[],
                               unsigned int size, char *targetname)
```

Υλοποίηση

Η συνάρτηση διατρέχει τον πίνακα συγκρίνοντας για κάθε εγγραφή το πεδίο name με την παράμετρο search. Για την 1η εγγραφή που η συνάρτηση strcmp θα επιστρέψει μηδέν επιστρέφει τη διεύθυνση της εγγραφής αυτής, δηλ

```
for(i=0; i<size; i++) {
    if(strcmp(students[i].name, targetname) == 0) {
        return &(students[i]);
    }
}
```

Το κυρίως πρόγραμμα (main)

Στη συνάρτηση main καλείστε να ορίσετε τον πίνακα μεγέθους MAX_STUDENTS και μία μεταβλητή τύπου μη προσημασμένου ακεραίου που αντιπροσωπεύει το μέγεθος του πίνακα αρχικοποιημένη σε 0 (αρχικά ο πίνακας είναι άδειος).

```
student_t students[MAX_STUDENTS];
unsigned int size = 0
```

Χρήση της συνάρτησης εισαγωγής:	`append_student_info(students, &size);`
Χρήση της συνάρτησης εκτύπωσης:	`for(i=0; i<size; i++)` `  print_student_info(students[i]);`

Χρήσης της συνάρτησης διαγραφής:

```
if( delete_student_info(students, &size, name) )
     printf("Remaining students %d\n", size);
else
     printf("No student found!\n");
```

Χρήσης της συνάρτησης αναζήτησης:

```
struct student *student_ptr;
student_ptr = search_student_info(students, size, name);
if(student_ptr)
     print_student_info(*student_ptr);
else
     printf("No student found!\n");
```

Έξοδος προγράμματος: Η έξοδος του προγράμματος πρέπει να είναι ίδια με την εκφώνηση.