Ασκήσεις Γ΄Ανακεφαλαίωσης

Για την κάθε μία από τις παρακάτω εργασίες:

(α) χρησιμοποιείστε ένα ξεχωριστό project – το κατάλληλο smProject που δίνεται,

(β) βάλτε σχόλια στον κώδικα που εξηγούν τα βήματα της επίλυσης και

(γ) δημιουργήστε μέσα στην smMain κώδικα που θα επιδεικνύει την καλή λειτουργία των συναρτήσεων.

Συμβουλές: Διαβάστε προσεκτικά δύο ή τρεις φορές την εκφώνηση. Επιλέξτε περιγραφικά και αυτοεξηγούμενα ονόματα μεταβλητών. Χρησιμοποιήστε καλή στοίχιση. Κατά τη συγγραφή του προγράμματος προσθέστε όσες `printf` χρειαστείτε για να βλέπετε τις τιμές των μεταβλητών ώστε να εντοπίζετε ευκολότερα τυχόν λάθη, πριν παραδώσετε όμως, αφαιρέστε τις "βοηθητικές" και κρατήστε μόνο όσες είναι απαραίτητες.

Σημείωση: Στον τελικό κώδικα μην χρησιμοποιήσετε την `printf` μέσα στις ζητούμενες συναρτήσεις.

Συμβουλές: Διαβάστε προσεκτικά την εκφώνηση. Επιλέξτε περιγραφικά ονόματα μεταβλητών. Χρησιμοποιήστε καλή στοίχιση (formatting) του κώδικα.

ΠΡΟΣΟΧΗ! Μην ξεχάσετε να κατεβάσετε και να χρησιμοποιήσετε τα αντίστοιχα smProject για την κάθε άσκηση!

ΠΡΟΣΕΞΤΕ ΟΠΩΣΔΗΠΟΤΕ ΤΑ ΠΑΡΑΚΑΤΩ

Ο τρόπος με τον οποίο πρέπει να υποβάλλετε ερωτήσεις περιγράφεται εδώ:

https://qna.c-programming.allos.gr/doku.php?id=qna:technical:questions

Ο τρόπος με τον οποίο πρέπει να υποβάλλετε τον κώδικα των εργασιών στο σύστημα υποβολής περιγράφεται εδώ:

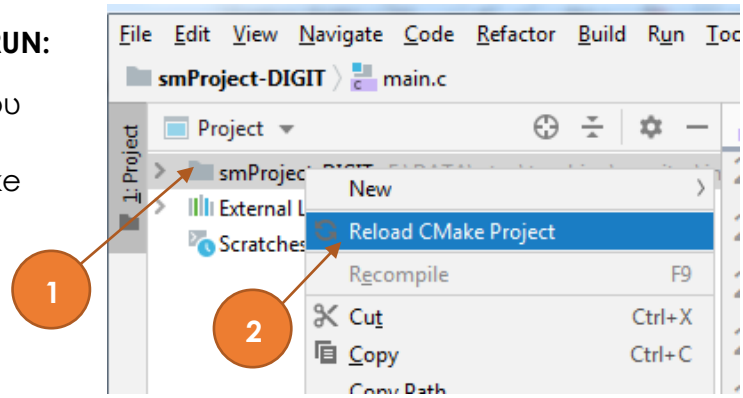
https://qna.c-programming.allos.gr/doku.php?id=qna:lesson:projects:how-to-submit

Οι έτοιμες συναρτήσεις της βιβλιοθήκης smLib που διατίθενται για χρήση στα smProjects, περιγράφονται (εκτός από τις διαφάνειες) και στο site ερωταποκρίσεων εδώ:

https://qna.c-programming.allos.gr/doku.php?id=qna:misc:sm-library

ΕΑΝ ΔΕΝ ΕΜΦΑΝΙΖΟΝΤΑΙ ΕΝΕΡΓΑ ΤΑ BUILD / RUN:

1. Κάνω **δεξί κλικ** πάνω στο όνομα του project και εμφανίζεται το μενού
2. Κάνω **απλό κλικ** στο Reload CMake Project, δηλαδή τη 2η επιλογή



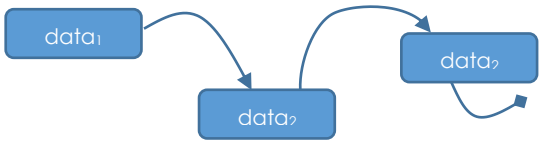
Εργασία Γ1 – Ελάχιστη τιμή λίστας

Βαθμός δυσκολίας: **1/3**

Όνομα smProject: **smProject-MIN-OF-LIST**

Περιγραφή

Δίνεται η δομή (struct)




```
typedef struct _node {
    struct _node *nextNode;
    double data;
} Node;
```

όπου κάθε μία τέτοια δομή (ας την πούμε κόμβο) περιέχει έναν pointer για μια επόμενη όμοια δομή (επόμενο κόμβο), δημιουργώντας έτσι μια αλυσίδα κόμβων. Ο τελευταίος κόμβος της αλυσίδας έχει τιμή του `nextNode` ίση με `NULL`. Καλείστε να γράψετε τη συνάρτηση:

```
double minOfList(Node *firstNode)
```

η οποία επιστρέφει **την ελάχιστη** τιμή του πεδίου `data` που θα βρεθεί στην αλυσίδα που ξεκινά με τον κόμβο `firstNode`. Εάν η αλυσίδα δεν έχει κανένα κόμβο, δηλαδή η `firstNode` είναι `NULL`, τότε θα πρέπει να επιστρέφει την τιμή $+10^{30}$ ως ένδειξη λάθους.