4η διάλεξη – βρόχοι for & while, do/while & switch

Για την κάθε μία από τις παρακάτω εργασίες:

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

(β) αν θέλετε δημιουργήστε, μέσα στην main, κώδικα που θα επιδεικνύει την καλή λειτουργία των συναρτήσεων.

Σημείωση: Στον τελικό κώδικα των ζητούμενων συναρτήσεων μην χρησιμοποιήσετε τις `smPrint` ή `smGrid` μέσα στις ζητούμενες συναρτήσεις.

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 `smPrint` 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.

ΠΡΟΣΟΧΗ! Μην ξεχάσετε να γράψετε τον κώδικα στην σελίδα της απλοποιημένης C στην αντίστοιχη εργασία για την κάθε άσκηση, πριν τον υποβάλλετε στο σύστημα υποβολής, ώστε να μπορέσετε να δείτε τα αποτελέσματα των σχετικών ελέγχων.

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

Εργασία 4α – Game of life , μέτρηση γειτόνων

Βαθμός δυσκολίας: **1/3**

Περιγραφή

Το Game of life (https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life) το γνωρίσαμε σε προηγούμενη άσκηση.

Στην άσκηση εκείνη θεωρούνταν ήδη υπολογισμένο το πλήθος των γειτονικών κελιών που είναι ζωντανά για κάθε κελί.

Με τη χρήση του grid της smLib δημιουργήστε μία συνάρτηση:

```
function countNeighbors ( x, y )
```

η οποία επιστρέφει το πλήθος των ζωντανών γειτόνων του κελιού x,y. Εάν το κελί x,y είναι εκτός του πλέγματος τότε πρέπει να επιστρέφει -1, ως ένδειξη λάθους.

Προσέξτε όμως τα κελιά που είναι στο όριο του πλέγματος, καθώς μερικά από τα γειτονικά τους είναι εκτός πλέγματος, αλλά δεν πρέπει βέβαια να επιστρέφεται -1 για αυτά.

Από την smLib θα χρειαστείτε τις συναρτήσεις: **`smgTest`**, **`smgWidth`**, **`smgHeight`**.

Στην main εάν θέλετε μπορείτε να φτιάξετε ένα πλέγμα να ανάψετε κάποια κελιά για να κάνετε τις δικές σας δοκιμές. Όμως στις ενσωματωμένες δοκιμές υπάρχουν έτοιμα πλέγματα (γι'αυτό δεν πρέπει και να καλέσετε την smGrid μέσα στη συνάρτησή σας) με αρκετές περιπτώσεις στις οποίες μπορεί – αν και δεν πρέπει – ο κώδικάς σας να αποτύχει.

Εργασία 4β – Οριζόντιες και κατακόρυφες

Βαθμός δυσκολίας: **2/3**

Περιγραφή

Καλείστε να γράψετε τις συναρτήσεις:

- `function` **`drawVLine`**`( x, y1, y2 )` αυτή πρέπει να σχεδιάζει στο grid μία κατακόρυφη γραμμή από το σημείο (x,y1) μέχρι και το (x,y2)
- `function` **`drawHLine`**`( y, x1, x2 )` αυτή πρέπει να σχεδιάζει στο grid μία οριζόντια γραμμή από το σημείο (x1,y) μέχρι και το (x2,y)
- `function` **`drawALine`**`( x, y, L, isV )` αυτή πρέπει να σχεδιάζει μία γραμμή που ξεκινά από το (x,y) και εκτείνεται δεξιά ή κάτω (αν το isVert είναι false ή true αντίστοιχα) με συνολικό μήκος L κελιά. Φυσικά το L πρέπει να είναι θετικό ή μηδέν.

Και οι 3 συναρτήσεις θα πρέπει να επιστρέφουν true εάν όλα πήγαν καλά, ενώ false εάν το ευθύγραμμο τμήμα που θα σχεδιάσουν βγαίνει έξω από τα όρια του grid, ή όταν το L είναι αρνητικό.

Στις υπόλοιπες περιπτώσεις θα πρέπει να λειτουργεί.

Η **`drawALine`** θα πρέπει να χρησιμοποιεί της **`drawHLine`** και **`drawVLine`** , ώστε να είναι απλή κατά το δυνατόν.

ΠΡΟΣΟΧΗ! Μην υποθέσετε ότι y1 < y2 ή x1 < x2 κλπ. Ο κώδικας θα πρέπει να λειτουργεί και για y1>=y2 και x1>=x2.

Τα tests σε διάφορα σημεία ελέγχουν το αποτέλεσμα των σχεδιασμών.
Εάν αποτύχει ένας τέτοιος έλεγχος, τότε δεν εκτελούνται τα επόμενα tests μέχρι να διορθωθεί το σφάλμα, ώστε να μπορείτε να βλέπετε το αποτέλεσμα των εντολών των tests και το πώς θα έπρεπε να είναι.
Δείτε το υπόμνημα κάτω από το πλέγμα.