Εργασία στο μάθημα Προχωρημένα Θέματα Προγραμματισμού Η/Υ

(έκδοση 1.0)

Δημιουργία ενός παίκτη του παιχνιδιού Proximity.

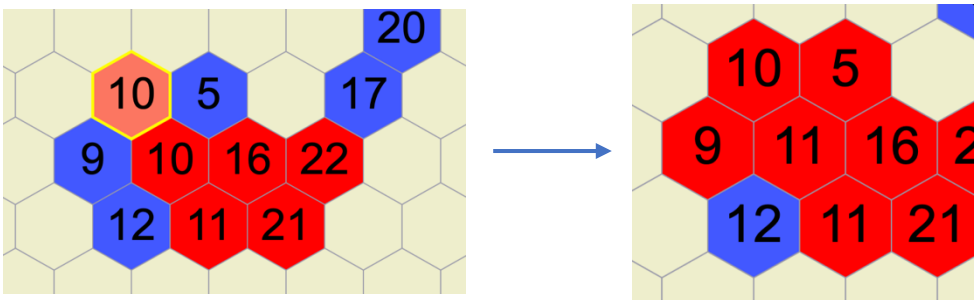
Ημερομηνία παράδοσης της εργασίας: **26 – 05 – 2024**
(50% τελικού βαθμού)

Τι είναι το Proximity?

Παιχνίδι στρατηγικής (https://www.mathsisfun.com/games/proximity.html). Υπάρχει ένα ταμπλό από εξάγωνα και δύο παίκτες, ο κόκκινος και ο μπλε. Σε κάθε γύρο ο παίκτης λαμβάνει έναν τυχαίο αριθμό, π.χ. x, από το 1 έως το 20 και τον τοποθετεί σε ένα ελεύθερο κελί του ταμπλό. Εάν το κελί αυτό είναι γείτονας με κελιά:

(i) του ίδιου χρώματος, τους αυξάνει την τιμή κατά 1 (μέγιστη τιμή το 20).
(ii) του άλλου χρώματος, τα καταλαμβάνει (δηλαδή τα αλλάζει χρώμα) εάν η τιμή τους είναι μικρότερη από x.

Ένα παράδειγμα φαίνεται στην Εικόνα 1.
Το παιχνίδι τελειώνει όταν δε μείνει κανένα ελεύθερο κελί στο ταμπλό. Κερδίζει ο παίκτης που έχει τους περισσότερους πόντους.



Εικόνα 1*. Ο κόκκινος τοποθετεί το 10. Τα μπλε κελιά με τιμές 9 και 5 καταλαμβάνονται ενώ το κελί κόκκινο 10 αυξάνεται κατά 1 και γίνεται 11.*

Ζητούμενα της εργασίας

1) Δημιουργείστε μια κλάση με όνομα *Cell* η οποία θα περιλαμβάνει τα παρακάτω:
   a. Instance variable "*value*" # Ακέραιος που δείχνει την τρέχουσα τιμή του κελιού. Η μέγιστη τιμή που μπορεί να πάρει ένα κελί είναι 20.
   b. Instance variable "*owner*" # Ακέραιος που δείχνει τον ιδιοκτήτη του κελιού. Μπορεί να πάρει τις τιμές 0 (δεν ανήκει σε κανένα παίκτη), 1 (κόκκινο), 2 (πράσινο), 3 (υποψήφιο για να το καταλάβει κάποιος παίκτης).
   c. Methods *getValue()*, *setValue(int)*, *getOwner()*, και *setOwner(int)*. Οι μέθοδοι set δίνουν τιμή στα αντίστοιχα instance variables ενώ οι μέθοδου get επιστρέφουν την τιμή των αντίστοιχων instance variables.
2) Δημιουργείστε μια κλάση με όνομα *ProximityXX* όπου το XX θα αντικατασταθεί με τον αριθμό της ομάδας σας (π.χ., Proximity05, Proximity12 κλπ.). Η κλάση αυτή θα περιλαμβάνει:

a. Instance variable "*pid*". Ακέραιος που δείχνει τον αριθμό του παίκτη (1 για κόκκινο, 2 για πράσινο)
b. Instance variables "length_X" and "length_Y". Ακέραιοι που δείχνουν το μέγεθος του ταμπλό.
c. Methods *setPid(int)* και *setBoardSize(int, int)*. Οι μέθοδοι θέτουν τιμή στα instance variables *pid* και *length_X* και *length_Y*, αντίστοιχα.
d. Method *getPlayerName()*. Η μέθοδος αυτή επιστρέφει ως string τα ονόματα των φοιτητών της ομάδας (π.χ. "Charisis_Papadopoulos_Ioannidis").
e. Method *findNeighbours(board)*.
   - Είσοδος: board (Cell list) – το ταμπλό στην τρέχουσα κατάσταση.
   - Έξοδος: temp_board (Cell list) – το ταμπλό σε μια προσωρινή κατάσταση.
   - Το ταμπλό αναπαρίσταται από μια λίστα που περιέχει αντικείμενα τύπου Cell. Η θέση του κελιού στη λίστα αντιστοιχεί σε συγκεκριμένη θέση στο ταμπλό. Η αντιστοιχία φαίνεται στην **Εικόνα 2**.
   - Το ταμπλό στην προσωρινή κατάσταση πρέπει να περιέχει ότι και το ταμπλό που δίνεται ως είσοδος. Επιπλέον, τα κελιά που είναι γείτονες των κατειλημμένων κελιών (ανεξαρτήτως χρώματος) πρέπει να πάρουν την τιμή *owner* = 3 και *value* = 0. Εάν δεν υπάρχει κατειλημμένο κελί, τότε η έξοδος πρέπει να είναι ίδια με την είσοδο. Προσοχή στις οριακές συνθήκες.
   - Το temp_board πρέπει να είναι αντίγραφο του board. Για το σκοπό αυτό πρέπει να χρησιμοποιηθεί η εντολή: *temp_board = board.copy()*.

Παράδειγμα 1

Έστω ότι παίζει ο πράσινος (owner = 2) και το τρέχον ταμπλό δίνεται από την παρακάτω λίστα:

```
board = [cell(0,0), cell(0,0), cell(15,1), cell(8,2), cell(0,0),
cell(0,0), cell(0,0), cell(0,0), cell(0,0), cell(0,0), cell(0,0),
cell(0,0), cell(0,0), cell(0,0), cell(0,0), cell(0,0), cell(0,0), …]
```

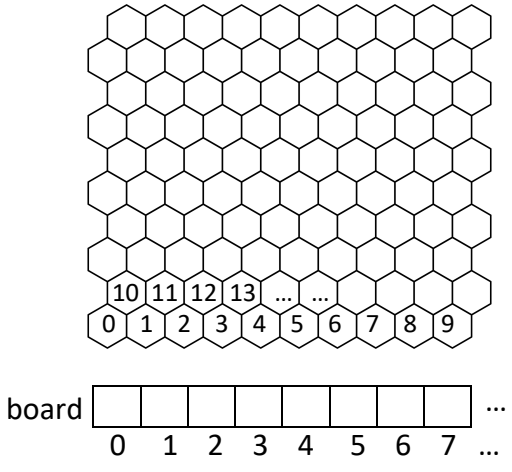
Δηλαδή, τα κελιά στις θέσεις 2 και 3 είναι κατειλημμένα.

Η συνάρτηση findNeighbours() πρέπει να επιστρέψει την παρακάτω λίστα:

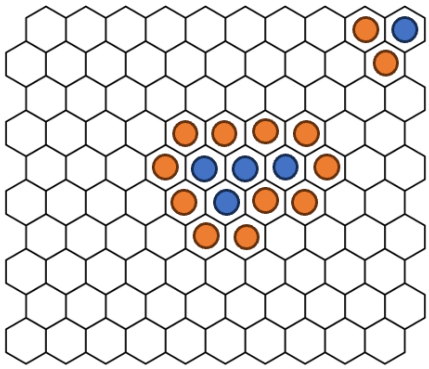
```
temp_board = [cell(0,0), cell(0,3), cell(15,1), cell(8,2), cell(0,3),
cell(0,0), cell(0,0), cell(0,0), cell(0,0), cell(0,0), cell(0,0),
cell(0,3), cell(0,3), cell(0,3), cell(0,0), cell(0,0), cell(0,0), …]
```

Δηλαδή, τα γειτονικά κελιά είναι αυτά που βρίσκονται στις θέσεις 1, 11, 12, 13 και 4.

Παράδειγμα 2: Στην Εικόνα 3 με 🔵 σημειώνονται τα κατειλημμένα κελιά και με 🟠 τα κελιά γείτονες που θα πρέπει να πάρουν τιμή *owner = 3*.



Εικόνα 2. Αρίθμηση κελιών του ταμπλό



Εικόνα 3. Τα γειτονικά κελιά που πρέπει να εντοπίσει η μέθοδος findNeighbours() είναι τα 🟠

f. Method *findBestPosition(tile_value, board)*.
   - Είσοδοι: (i) *tile_value* (Int) - ο αριθμός που πρέπει να τοποθετηθεί σε κάποιο κελί του ταμπλό; (ii) *board* (Cell list) - το ταμπλό στην τρέχουσα κατάσταση.
   - Έξοδος: (i) *best_tile_position* (Int) - η θέση του κελιού (σύμφωνα με την αρίθμηση της Εικόνας 2) στην οποία θα τοποθετηθεί ο αριθμός.
   - Η μέθοδος *findBestPosition()* πρέπει να καλεί τη μέθοδο *findNeighbours()*, να αξιολογεί τις πιθανές θέσεις που μπορεί να τοποθετηθεί ο νέος αριθμός και να τον τοποθετεί στην καταλληλότερη θέση. Εάν είναι ο πρώτος αριθμός που τοποθετείται στο ταμπλό τότε πρέπει να τοποθετηθεί στο κεντρικό κελί του ταμπλό. Εάν καμία από τις γειτονικές θέσεις (που βρίσκει η *findNeighbours()* δεν είναι συμφέρουσα, τότε ο αριθμός μπορεί να τοποθετηθεί σε οποιοδήποτε ελεύθερο κελί του ταμπλό.

g. Method *findMyNeighbours(tile_position)*.
   - Είσοδος: tile_position (Int) - Η θέση ενός κελιού στα ταμπλό (όπως φαίνεται στην Εικόνα 2).
   - Έξοδος: neighbours_positions (Int list) - Οι θέσεις των γειτονικών κελιών (ανεξάρτητα με το εάν είναι κενά ή όχι) σε λίστα. Προσοχή στις οριακές συνθήκες (άκρα του ταμπλό).
   - *Παράδειγμα*: Σύμφωνα με την Εικόνα 2, η findMyNeighbours(2) πρέπει να επιστρέψει τη λίστα [1, 3, 11, 12].

h. Method *placeTile(new_piece, board)*.
   - Είσοδοι: (i) new_piece (Int) - ο αριθμός που πρέπει να τοποθετηθεί σε κάποιο κελί του ταμπλό, (ii) board (Cell list) - το ταμπλό στην τρέχουσα κατάσταση.
   - Έξοδος: (i) board_updated (Cell list) - το ταμπλό στη νέα κατάσταση.
   - Η μέθοδος *placeTile()* τοποθετεί τον αριθμό στο κατάλληλο κελί (πρέπει να χρησιμοποιηθεί η *findBestPosition()*) και έπειτα ανανεώνει τις τιμές και το χρώμα των γειτονικών κελιών (πρέπει να χρησιμοποιηθεί η *findMyNeighbours()*) με βάση τους κανόνες του Proximity.

3) Δημιουργείστε κυρίως πρόγραμμα το οποίο να αποδεικνύει την καλή λειτουργία των παραπάνω.

Παραδοτέα

1. Ένα αρχείο με όνομα **ProximityXX_Player.py**, όπου το X θα αντικατασταθεί από τον αριθμό της ομάδας σας. Το αρχείο αυτό πρέπει να περιλαμβάνει τις κλάσεις *Cell* και *ProximityXX* που δημιουργήσατε καθώς και το κυρίως πρόγραμμα που αποδεικνύει την σωστή λειτουργία των κλάσεων.
2. Σύντομη γραπτή αναφορά με όνομα **ProximityXX_Report.pdf**, στην οποία πρέπει να περιέχονται τα ακόλουθα:
   a. Σύντομη περιγραφή του προβλήματος και των αλγορίθμων που υλοποιήσατε.
   b. Σύντομη περιγραφή της λογικής με την οποία η μέθοδος *findBestPosition()* αξιολογεί τις διαθέσιμες θέσεις και τοποθετεί ένα αριθμό στο ταμπλό.
   c. Αποτελέσματα που να δείχνουν την ορθή λειτουργία του κώδικα (π.χ. screenshots) με επεξηγήσεις και σχόλια.
   d. Πίνακας όπου να φαίνεται με τι ασχολήθηκε κάθε μέλος της ομάδας και σε τι ποσοστό.

   e. Τον κώδικα (όχι με τη μορφή εικόνας)
3. Ένα .zip αρχείο με όνομα XX_Επώνυμο1_Επώνυμο2_Επώνυμο3.zip, όπου XX είναι ο αριθμός ομάδας (π.χ., 05_Papadopoulos_Papageorgiou_Oikonomou.zip) το οποίο θα περιέχει τα ProximityXX_Player.py και ProximityXX_Report.pdf. Το zip αρχείο πρέπει να κατατεθεί στο elearning έως 26-05-2024.

Προσοχή

1) Τα ονόματα των κλάσεων, μεθόδων κλπ. πρέπει να είναι ακριβώς όπως στην εκφώνηση της εργασίας.
2) Τα ονόματα των παραδοτέων πρέπει να είναι ακριβώς όπως στην εκφώνηση της εργασίας.
3) Ο κώδικας πρέπει να συνοδεύεται από πλήρη τεκμηρίωση (σχόλια). Στην αρχή του αρχείου ProximityXX_Player.py πρέπει να υπάρχει επικεφαλίδα σε μορφή σχολίων με τα στοιχεία σας (ονομεταπώνυμα, ΑΕΜ, email). Επίσης, πριν από κάθε μέθοδο (method) πρέπει να υπάρχει επικεφαλίδα με τη μορφή σχολίων όπου θα παρουσιάζεται μια σύντομη περιγραφή της λειτουργικότητας της και θα περιγράφονται οι μεταβλητές τους
4) Οι αναφορές και ο κώδικας θα περάσουν από εργαλείο ελέγχου λογοκλοπής. Σε περίπτωση λογοκλοπής η εργασία θα μηδενίζεται.