Πανεπιστήμιο Ιωαννίνων, Τμήμα Μηχανικών Η/Υ και Πληροφορικής
ΜΥΥ205 - ΠΛΥ212: Τεχνικές Αντικειμενοστρεφούς Προγραμματισμού
ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2016-17
Δεύτερη Άσκηση
Ημερομηνία Παράδοσης: 2 Μαΐου 2017, 3 μ.μ.

Η εργασία αυτή μετράει 10% του συνολικού βαθμού του μαθήματος και είναι η δεύτερη από τις τρεις που θα δοθούν συνολικά.

Στόχος της εργασίας είναι να φτιαχτεί ένα πρόγραμμα το οποίο θα βρίσκει τη λύση ενός λαβύρινθου ή θα διαπιστώνει ότι δεν υπάρχει λύση. Ο λαβύρινθος μοντελοποιείται με τη βοήθεια ενός δισδιάστατου δυαδικού πίνακα. Σε κάθε θέση του πίνακα υπάρχει είτε true, το οποίο σημαίνει ότι επιτρέπεται η πρόσβαση σε αυτή τη θέση, είτε false. Το σημείο εισόδου στο λαβύρινθο είναι στην 1η γραμμή και στήλη, δηλ. η θέση (0,0) και το σημείο εξόδου είναι στην τελευταία γραμμή και στήλη, δηλ. η θέση (αρ. γραμμών - 1, αρ. στηλών - 1). Τα σημεία εισόδου και εξόδου πρέπει να είναι οπωσδήποτε true, αλλιώς δεν υπάρχει λύση στο λαβύρινθο. Από κάθε θέση επιτρέπεται να μεταβούμε σε μία από τις 4 γειτονικές θέσεις (οριζόντια και κάθετα, αλλά όχι διαγώνια), εφόσον η θέση αυτή επιτρέπει την πρόσβαση (δηλ. είναι true). Για παράδειγμα, στο Σχήμα 1 που ακολουθεί, στα αριστερά δίνεται ένας λαβύρινθος, όπου οι προσβάσιμες θέσεις μαρκάρονται με 0, ενώ οι μή προσβάσιμες με 1. Στα δεξιά βλέπετε ένα μονοπάτι μαρκαρισμένο με *, το οποίο αποτελεί λύση στο λαβύρινθο.

0	1	0	1	0	1
0	1	0	1	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	1	0

*	1	0	1	0	1
*	1	0	1	0	0
*	1	*	*	*	*
*	*	*	0	1	*
0	0	0	0	1	*

Σχήμα 1: Παράδειγμα λαβυρίθνου και λύσης

Για την εργασία αυτή θα υλοποιήσετε τρεις κλάσεις. Η πρώτη κλάση θα είναι η Maze η οποία έχει σαν πεδίο έναν λαβύρινθο με τη μορφή δισδιάστατου δυαδικού πίνακα. Η Maze πρέπει να έχει μια μέθοδο solve() η οποία επιστρέφει true αν υπάρχει λύση στο λαβύρινθο και false αλλιώς. Η solve χρησιμοποιεί μια στοίβα για να λύσει το πρόβλημα ως εξης:

- Κάθε στοιχείο της στοίβας περιέχει ένα κελλί του πίνακα και τις νόμιμες κινήσεις μετάβασης από αυτό το κελλί.

- Ξεκινώντας από το κελλί (0,0), το οποίο πρέπει να είναι true, εξετάζουμε το τρέχον κελλί και βρίσκουμε τις δυνατές μεταβάσεις στους γείτονές του. Δοκιμάζουμε την επόμενη δυνατή μετάβαση βάζοντας το κελλί μας στη στοίβα και κάνοντας το αντίστοιχο γειτονικό κελλί τρέχον. Αν φτάσουμε στην έξοδο του λαβυρίνθου τερματίζουμε την εύρεση επιστρέφοντας true. Αν φτάσουμε σε ένα κελλί από το οποίο δεν μπορούμε να μεταβούμε σε νέο κελλί (αδιέξοδο), επιστρέφουμε στο προηγούμενο στοιχείο της στοίβας και δοκιμάζουμε την επόμενη κίνηση. Η στοίβα μας βοηθάει στο πισωγύρισμα και στο να θυμόμαστε ποια θα είναι η επόμενη κίνηση από το προηγούμενο κελλί.

Το Σχήμα 2 δείχνει παραδείγματα χρήσης της στοίβας για να λύσουμε 2 προβλήματα. Τα λευκά τετράγωνα αντιστοιχούν σε θέσεις όπου επιτρέπεται η πρόσβαση (true) ενώ τα σκούρα σε θέσεις όπου απαγορεύεται (false). Ο λαθύρινθος στα αριστερά έχει λύση, ενώ αυτός στα δεξιά όχι.

Εκτός από την solve, υλοποιήστε τη μέθοδο toString η οποία θα γράφει σε ένα αλφαριθμητικό για κάθε θέση του λαβυρίνθου 0 ή 1 ανάλογα με το αν είναι προσβάσιμη ή όχι. Επίσης υλοποιήστε μια μέθοδο printSolution η οποία θα τυπώνει στην έξοδο το λαβύρινθο, μαρκάροντας το μονοπάτι της λύσης με *. Για τη λειτουργία της, χρειάζεται να γνωρίζετε τη στοίβα, την οποία μπορείτε να κρατήσετε σε μια private μεταβλητή της Maze.

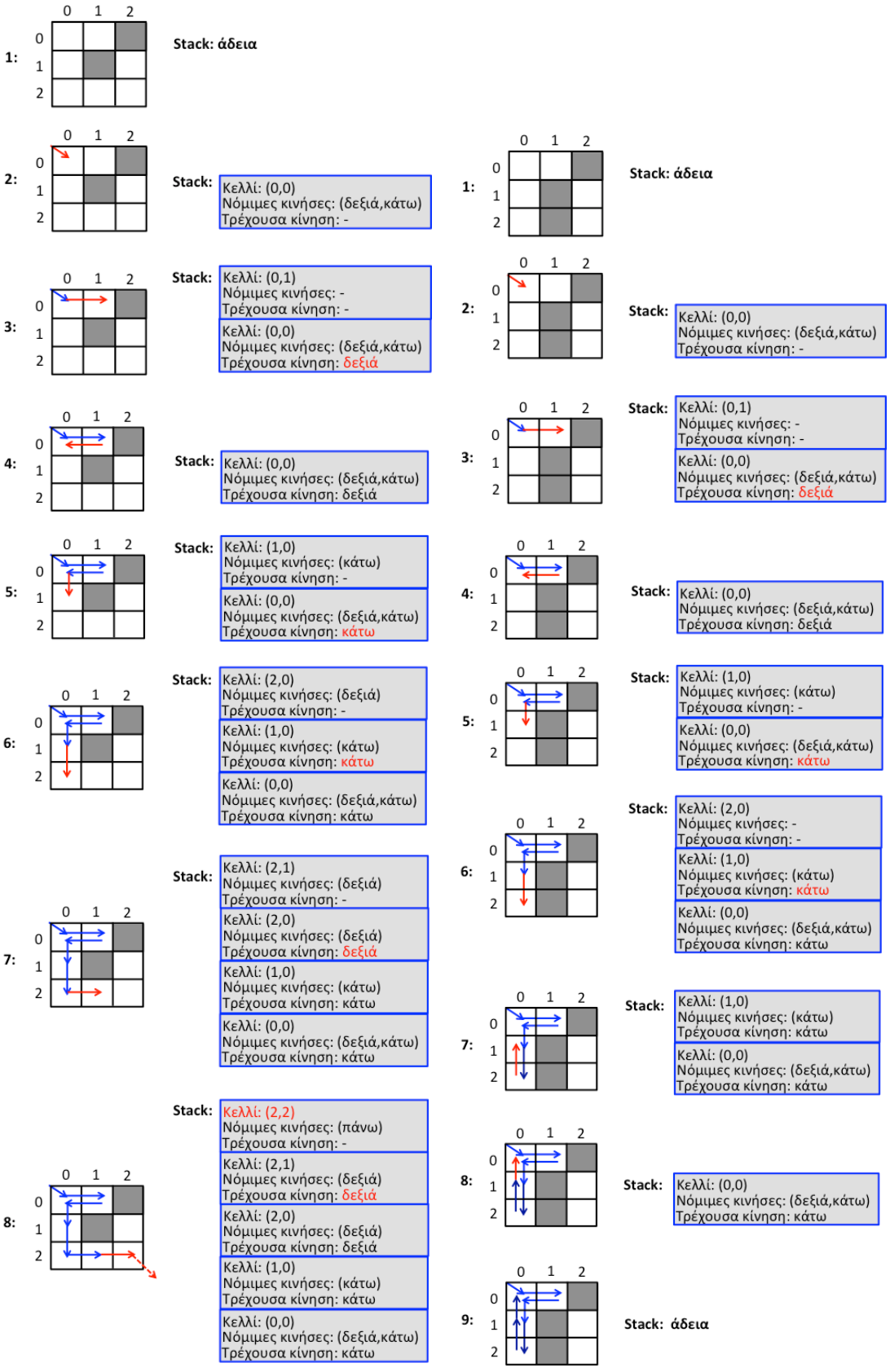
Η 2η κλάση που θα χρειαστείτε είναι η StackOfStates, η οποία είναι μιά στοίβα που αποθηκεύει αντικείμενα τύπου State (η State είναι η 3η κλάση). Εκτός από τις μεθόδους pop, push, isEmpty, φτιάξτε και μια μέθοδο inStack, η οποία θα παίρνει δύο όρισματα: γραμμή και στήλη και θα επιστρέφει true, άν υπάρχει στοιχείο στη στοίβα που να αντιστοιχεί στο κελλί στη συγκεκριμένη γραμμή και στήλη. Η μέθοδος αυτή μας βοηθάει στο να αποφύγουμε να βάλουμε 2 states που αντιστοιχούν στο ίδιο κελλί στη στοίβα για να μην κάνουμε κύκλους στα κελλιά κατά την εύρεση.

Η 3η κλάση (State) δημιουργείται για τα στοιχεία της στοίβας. Το κάθε στοιχείο αντιστοιχεί σε ένα κελλί του λαβυρίνθου και περιγράφεται από τη γραμμή και τη στήλη του, από μία ακολουθία από νόμιμες κινήσεις σε άλλα κελιά (πάνω, κάτω, αριστερά, δεξιά) και από ένα πεδίο που μαρκάρει την τρέχουσα κίνηση που έχουμε ακολουθήσει από αυτό το κελλί. Αν π.χ. το κελλί βρίσκεται στην πρώτη γραμμή, δεν επιτρέπεται η κίνηση προς τα πάνω. Αν το κελλί στα δεξιά έχει false, δεν επιτρέπεται η κίνηση προς τα δεξιά, κλπ.

Στις κλάσεις σας προσθέστε τις μεθόδους που χρειάζονται για να λυθεί το πρόβλημα. Π.χ. για τη μέθοδο Maze.printSolution μπορεί να χρειάζεται να προσθέστε μεθόδους toString στις άλλες κλάσεις.

Δίνεται μια κλάση MazeTest με το main πρόγραμμα για έλεγχο. Επίσης ένα αρχείο για το τι πρέπει να τυπώνει η main. Για τον έλεγχο του προγράμματός σας, μπορεί να χρησιμοποιηθεί άλλη main συνάρτηση.

Στην υλοποίηση των κλάσεων σας δεν θα πρέπει να έχετε public πεδία. Βαθμοί θα αφαιρεθούν για προγράμματα που δεν είναι καλά γραμμένα, δηλαδή δεν είναι σωστά στοιχισμένα ή δεν έχουν καλά επιλεγμένα ονόματα μεταβλητών ώστε να διαβάζονται εύκολα. Προγράμματα που δεν περνάνε από compiler δεν θα πάρουν πάνω από τους μισούς βαθμούς. Μετά την ημερομηνία παράδοσης δεν θα γίνονται δεκτές οι υποβολές των εργασιών.



(α) παράδειγμα που καταλήγει σε λύση (β) παράδειγμα χωρίς λύση

Σχήμα 2: Παραδείγματα εκτέλεσης προγράμματος