ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΠΛ131 Αρχές Προγραμματισμού

Ακαδημαϊκό Έτος 2017/18 – Χειμερινό Εξάμηνο

ΤΕΛΙΚΗ ΕΞΕΤΑΣΗ ΕΞΑΜΗΝΟΥ

ΗΜΕΡΟΜΗΝΙΑ: 11 Δεκεμβρίου 2017
ΔΙΑΡΚΕΙΑ: 12:30μμ – 3:30μμ
ΑΙΘΟΥΣΑ: Κτήριο ΧΩΔ02, Αίθουσα Β205
ΔΙΔΑΣΚΟΥΣΑ: Ελπίδα Κεραυνού-Παπαηλιού

Απαντήστε και τις τρεις ερωτήσεις
Κάθε ερώτηση λαμβάνει 33 μονάδες.

Ερώτηση 1

Το αντικείμενο CMemory, το οποίο αντιπροσωπεύει μια μνήμη στην οποία αποθηκεύονται χαρακτήρες, ορίζεται μερικώς πιο κάτω. Ο ορισμός αυτός χρειάζεται να συμπληρωθεί με τους ορισμούς των σχετικών μεθόδων αναφοράς όπως εξηγείται στα σχόλια που δίνονται.

```
public class CMemory {

  private int size; // το μέγεθος της μνήμης
  private char[] content; // το περιεχόμενο της μνήμης
  private boolean[] status; // παράλληλος πίνακας με τον πίνακα
                           // content που περιγράφει την κατάσταση
                           // των κυψελίδων της μνήμης – αν μια
                           // κυψελίδα είναι ελεύθερη, τότε το
                           // status της είναι false, διαφορετικά
                           // είναι true
  private int[][] holes; // καταγράφονται οι «τρύπες» (holes) της
                         // μνήμης, δηλαδή τα συνεχόμενα τμήματα
                         // ελεύθερων κυψελίδων – η κάθε «τρύπα»
                         // προσδιορίζεται από που αρχίζει και που
                         // τελειώνει
  private int numHoles; // το πλήθος των «τρυπών» της μνήμης
```

```
// Ο βασικός κατασκευαστής, ο οποίος ήδη ορίζεται. Η μοναδική
// παράμετρός του αντιπροσωπεύει το μέγεθος της μνήμης σε αριθμό
// χαρακτήρων
public CMemory (int s){
  size = s;
  content = new char[size]; // δημιουργείται ο χώρος για το
                            // περιεχόμενο της μνήμης
  status = new boolean[size];
  for (int i = 0; i < size; i++)
      status[i] = false; // αρχικά κάθε κυψελίδα είναι ελεύθερη
}

// Εναλλακτικός τρόπος κατασκευασμού μιας μνήμης, μέσω αντιγραφής
// κάποιας άλλης μνήμης που αποτελεί την παράμετρο του δεύτερου
// αυτού, copy constructor – NA ΟΡΙΣΘΕΙ
public CMemory (CMemory cm){
 . . . . . .
}

// Επιστρέφει το μέγεθος της μνήμης
public int getSize(){return size;}

// Επιστρέφει την κατάσταση της ι-οστής κυψελίδας της μνήμης,
// δηλαδή false αν είναι ελεύθερη, ή true αν είναι καταχωρημένη
public boolean getStatus(int i){return status[i];}

// Διπλασιάζει το μέθεθος της μνήμης, δηλαδή αντιγράφει τα
// περιεχόμενα και την κατάστασή τους σε διπλάσιου μεγέθους πίνακες
// όπου όλες οι επιπρόσθετες κυψελίδες είναι ελεύθερες – NA ΟΡΙΣΘΕΙ

public void doubleM (){
  . . . . . .
}

// Η ιδιωτική μέθοδος αναφοράς, computeHoles(), υπολογίζει τις
// τρέχουσες «τρύπες» της μνήμης, αναθέτει δηλαδή τιμές στα πεδία
// holes και numHoles – NA ΟΡΙΣΘΕΙ

private void computeHoles(){
  . . . . . .
}

// Επιστρέφει το συνολικό ελεύθερο χώρο της μνήμης – NA ΟΡΙΣΘΕΙ

public int freeSpace(){
  . . . . . .
}

// Επιστρέφει το τρέχον ποσοστό της μνήμης που είναι κατανεμημένο.
// Η μέθοδος είναι ήδη ορισμένη.

public double usage(){
   return (100.0 * (size - freeSpace()))/size;
}

// Ελευθερώνει το τμήμα της μνήμης μήκους len κυψελίδων το οποίο
// αρχίζει από την index κυψελίδα – NA ΟΡΙΣΘΕΙ
public boolean deallocate (int index, int len){
   . . . . . .
}
```

```
  // Αποθηκεύει τη συμβολοσειρά s στην πρώτη σε σειρά «τρύπα» της
  // μνήμης, της οποίας το μέγεθος είναι τουλάχιστο ίσο με το μήκος
  // της συμβολοσειράς. Εάν η επιλεγείσα «τρύπα» είναι μεγαλύτερη
  // από το μήκος της συμβολοσειράς το επιπρόσθετο τμήμα της
  // παραμένει ελεύθερο, δημιουργώντας έτσι μια μικρότερου
  // μεγέθους «τρύπα». Η μέθοδος επιστρέφει τη διεύθυνση της
  // κυψελίδας από την οποία αρχίζει η αποθήκευση της s. Εάν καμία
  // «τρύπα» της μνήμης δεν είναι αρκετά μεγάλη για την αποθήκευση
  // της s, τότε η s δεν αποθηκεύεται στη μνήμη και η μέθοδος απλά
  // επιστρέφει την εικονική τιμή -1 ΝΑ ΟΡΙΣΘΕΙ

  public int allocate(String s){
    . . . . . .
  }


  // Συμπυκνώνει τα ελεύθερα τμήματα της μνήμης σε ένα ενιαίο
  // ελεύθερο τμήμα στο κάτω μέρος της μνήμης – ΝΑ ΟΡΙΣΘΕΙ

  public void compact (){
     . . . . . .
  }


  // Η μέθοδος toString() για την παρουσίαση της κατάστασης του
  // περιεχομένου της μνήμης είναι ήδη ορισμένη – η λειτουργία της
  // μεθόδου επιδεικνύεται πιο κάτω

  public String toString(){
    String s = "\nThe memory has " + freeSpace() +
               " free cells as follows:\n";
    computeHoles();
    for (int i = 0; i < numHoles; i++){
      s += "\tFrom cell " + holes[i][0] + " to cell " +
           holes[i][1] + " inclusive\n";
    }
    s += "\n\nIts allocated and free regions are:\n";
    for (int i = 0; i < size; i++){
     if (i != 0){
       if (status[i] != status[i-1]) s += "\n";
     }
     if (status[i]) s += "a";
     else s += "f";
    }
    return s + "\nIts usage is " + usage() + "%\n";
 }

}
```

Έστω ότι έχουμε κάποια μνήμη χαρακτήρων μεγέθους 20 κυψελίδων, την cm. Μετά από διάφορες αποθηκεύσεις γραμματοσειρών και ελευθερώσεις χώρων της, η παρουσίαση της κατάστασης του περιεχομένου της, μέσω της εντολής System.out.print(cm), όπου αυτόματα ενεργοποιείται η μέθοδος toString(), θα μπορούσε να ήταν η ακόλουθη, όπου κατανεμημένη (allocated) κυψελίδα αναφέρεται ως a και ελεύθερη (free) κυψελίδα ως f:

```
The memory has 12 free cells as follows:
      From cell 3 to cell 5 inclusive
```

```
      From cell 11 to cell 19 inclusive


Its allocated and free regions are:
aaa
fff
aaaaa
fffffffff
Its usage is 40.0%
```

Έστω ότι στη συνέχεια γίνεται συμπύκνωση των ελεύθερων χώρων της cm σε μια ενιαία «τρύπα» στο κάτω μέρος της, μέσω της εντολής cm.compact(). Η εκ νέου εκτέλεση της εντολής System.out.print(cm) παρουσιάζει τη νέα κατάσταση του περιεχομένου της ως ακολούθως:

```
The memory has 12 free cells as follows:
      From cell 8 to cell 19 inclusive


Its allocated and free regions are:
aaaaaaaa
ffffffffffff
Its usage is 40.0%
```

Ερώτηση 2

Κατασκευάστε το πρόγραμμα MCAssess.java το οποίο βαθμολογεί τις απαντήσεις φοιτητών σε εξετάσεις όπου οι ερωτήσεις είναι πολλαπλών επιλογών (multiple choice questions) και επίσης παρουσιάζει κάποια στοιχεία για τις ερωτήσεις όπως εξηγείται στη συνέχεια.

Καταρχάς το πρόγραμμα εισαγάγει τα στοιχεία προς επεξεργασία από κάποιο αρχείο κειμένου μέσω file redirection. Το ακόλουθο αρχείο Marks.txt είναι ένα τέτοιο αρχείο. Οι πρώτοι δύο αριθμοί στο αρχείο αντιπροσωπεύουν το πλήθος των φοιτητών και το πλήθος των ερωτήσεων σε αυτή τη σειρά. Στο συγκεκριμένο παράδειγμα η εξέταση αφορά 20 φοιτητές και αποτελείται από 5 ερωτήσεις.

Στη συνέχεια, για κάθε ερώτηση, από την πρώτη μέχρι την τελευταία δίνεται ο αριθμός των επιλογών που είχε ο φοιτητής και ποια από αυτές τις επιλογές είναι η ορθή απάντηση στην ερώτηση. Στο πιο κάτω παράδειγμα η ερώτηση 1 έχει 4 επιλογές όπου η ορθή απάντηση είναι η επιλογή 1, ενώ η ερώτηση 5 έχει 3 επιλογές όπου η ορθή απάντηση είναι η επιλογή 3.

Ακολούθως δίνονται για κάθε φοιτητή οι απαντήσεις του στις εν λόγω ερωτήσεις, με τη σειρά των ερωτήσεων. Επισημαίνεται, ότι ένας φοιτητής δύναται να μην απαντήσει κάποια ερώτηση. Σε τέτοιες περιπτώσεις εμφανίζεται ο αριθμός 0. Στο συγκεκριμένο παράδειγμα ο φοιτητής 2 επέλεξε να μην απαντήσει τις ερωτήσεις 1 και 4.

Αρχείο Marks.txt

```
20 5
4 1 4 3 5 5 3 2 3 3
1 3 4 1 3
0 3 5 0 1
1 3 5 2 3
```

```
1 3 5 0 2
4 3 5 1 2
1 0 5 0 3
3 2 4 0 2
1 3 4 1 3
1 1 2 2 3
3 2 5 2 3
2 2 2 2 3
0 3 5 0 3
1 3 0 0 3
1 3 5 0 3
4 4 5 3 3
1 1 1 1 1
1 3 4 2 3
1 3 0 2 3
1 3 5 0 0
0 0 5 2 3
```

Η βαθμολογία ενός φοιτητή υπολογίζεται ως εξής: Για κάθε ορθή απάντηση λαμβάνει δύο μονάδες. Για κάθε λάθος απάντηση του αφαιρείται μία μονάδα. Για τυχόν ερωτήσεις που δεν απάντησε, δεν λαμβάνει, ούτε και χάνει τίποτα. Το output του προγράμματος για τα πιο πάνω στοιχεία εισόδου είναι το ακόλουθο:

$java MCAssess < Marks.txt

```
Student 1 mark is 4
Student 2 mark is 3
Student 3 mark is 10
Student 4 mark is 5
Student 5 mark is 1
Student 6 mark is 6
Student 7 mark is -4
Student 8 mark is 4
Student 9 mark is 4
Student 10 mark is 4
Student 11 mark is 1
Student 12 mark is 6
Student 13 mark is 6
Student 14 mark is 8
Student 15 mark is 1
Student 16 mark is -2
Student 17 mark is 7
Student 18 mark is 8
Student 19 mark is 6
Student 20 mark is 6

Question Statistics
Question  Correct  Wrong  Don't know
  1          12     5     3
  2          12     6     2
  3          11     7     2
  4          7      5     8
  5          14     5     1

Most difficult questions: 4
7 students answered correctly the above
Easiest questions: 5
14 students answered correctly the above
```

```
Histogram for Question 1
0 ---
1 ++++++++++++
2 -
3 --
4 --

Histogram for Question 2
0 --
1 --
2 ---
3 ++++++++++++
4 -

Histogram for Question 3
0 --
1 -
2 --
3
4 ----
5 ++++++++++

Histogram for Question 4
0 --------
1 ----
2 +++++++
3 -

Histogram for Question 5
0 -
1 --
2 ---
3 ++++++++++++++
```

Εκτός από τις βαθμολογίες των φοιτητών, το πρόγραμμα δίνει κάποιες στατιστικές για κάθε ερώτηση, συγκεκριμένα πόσοι φοιτητές την απάντησαν ορθά, πόσοι την απάντησαν λάθος και πόσοι δεν την απάντησαν καθόλου. Επιπρόσθετα αναφέρει ποιες ερωτήσεις ήταν οι πιο δύσκολες, δηλαδή αυτές που απαντήθηκαν ορθά από το μικρότερο αριθμό φοιτητών, καθώς και ποιες ερωτήσεις ήταν οι ευκολότερες, δηλαδή αυτές που απαντήθηκαν ορθά από το μεγαλύτερο αριθμό φοιτητών. Τέλος το πρόγραμμα παρουσιάζει το ιστόγραμμα για την κάθε ερώτηση στο οποίο φαίνεται το πλήθος των φοιτητών που δεν την απάντησαν, καθώς και το πλήθος των φοιτητών για κάθε μια από τις επιλογές που είχαν για τη συγκεκριμένη ερώτηση. Για την ορθή επιλογή η σειρά που απεικονίζει το σχετικό πλήθος αναγράφεται με το χαρακτήρα +, ενώ οι υπόλοιπες σειρές με το χαρακτήρα -.

Ερώτηση 3

Η κλάση Robot αντιπροσωπεύει μια κατηγορία αντικειμένων ρομπότ. Η κλάση ορίζεται μερικώς και χρειάζεται να συμπληρωθεί όπως εξηγείται στη συνέχεια. Καταρχάς ένα τέτοιο ρομπότ τοποθετείται σε ένα τετραγωνικό πλέγμα και κινείται με τυχαίο τρόπο όπου σε κάθε σημείο οι επιτρεπτές κινήσεις είναι ένα βήμα πάνω, κάτω, δεξιά ή αριστερά. Επιπρόσθετα, υπάρχουν οι ακόλουθοι περιορισμοί:

- Δεν μπορεί να περάσει ξανά από κάποιο σημείο που έχει ήδη περάσει.

- Αν φτάσει σε κάποιο ακραίο σημείο του πλέγματος, τότε σταματά και θεωρείται ότι έχει δραπετεύσει.
- Αν δεν μπορεί να κινηθεί από το μη-ακραίο σημείο που βρίσκεται επειδή έχει ήδη περάσει από όλα τα δυνατά σημεία, τότε παραμένει στάσιμο και παγιδευμένο.
- Αν μετακινηθεί σε σημείο που βρίσκεται κάποιο άλλο ρομπότ, τότε γίνεται σύγκρουση και κάθε συγκρουόμενο ρομπότ καταστρέφεται, παραμένοντας στο σημείο της σύγκρουσης. Το ίδιο συμβαίνει αν ένα ρομπότ είναι στάσιμο, είτε επειδή δραπέτευσε, είτε επειδή παγιδεύτηκε και κάποιο άλλο ρομπότ συγκρούεται μαζί του.

Συμπληρώστε την κλάση Robot.

```
import java.awt.Color;

public class Robot{
  private String name; // το όνομα του
  private int Gsize; // το μέγεθος του τετραγωνικού πλέγματος στο
                     // οποίο βρίσκεται
  private int xpos;    // η x συντεταγμένη της τρέχουσας θέσης του
  private int ypos;    // η y συντεταγμένη της τρέχουσας θέσης του
  private boolean alive; // είναι ζωντανό; αρχικά κάθε ρομπότ είναι
                         // ζωντανό
  private boolean escaped; // έχει δραπετεύσει, δηλαδή βρίσκεται σε
                           // ακραίο σημείο του πλέγματος;
  private boolean trapped; // είναι παγιδευμένο;
  private int[][] path;   // η διαδρομή που έχει διανύσει, δηλαδή η
                          // ακολουθία των συντεταγμένων των σημείων
                          // από τα οποία έχει περάσει – το αρχικό
                          // σημείο δίνεται στην πρώτη σειρά του path
                          // και το τρέχον σημείο που βρίσκεται
                          // δίνεται στην σειρά plen-1 του path
  private int plen; // το μήκος της μέχρι τώρα διαδρομής του
  private Color col; // το χρώμα του, το οποίο παράγεται τυχαία αλλά
                     // πρέπει να είναι συμβατό με το άσπρο (WHITE)


  // Ο μοναδικός κατασκευαστής λαμβάνει το όνομα του ρομπότ και τις
  // συντεταγμένες της αρχικής του θέσης, καθώς και το μέγεθος του
  // πλέγματος και αρχικοποιεί ανάλογα τα σχετικά πεδία του ρομπότ.
  // Μόνο ζωντανά ρομπότ κατασκευάζονται. ΝΑ ΟΡΙΣΘΕΙ

  public Robot(String n, int x, int y, int size){
      . . . . . .
  }

  // Είναι ζωντανό;
  public boolean isAlive(){return alive;}

  // Έχει δραπετεύσει;
  public boolean hasEscaped(){return escaped;}

  // Είναι παγιδευμένο;
  public boolean isTrapped(){return trapped;}

  // Το ρομπότ καταστρέφεται
  public void destroyed (){
     alive = false;
  }
```

```
  // Επιστρέφει την τρέχουσα θέση του ρομπότ. ΝΑ ΟΡΙΣΘΕΙ
  public int[] getPos(){
    . . . . . .
  }

 // Επιστρέφει την αρχική θέση του ρομπότ. ΝΑ ΟΡΙΣΘΕΙ
 public int[] getStart(){
   . . . . . .
 }


// Έχει συγκρουσθεί με το έτερο ρομπότ r; ΝΑ ΟΡΙΣΘΕΙ
public boolean collides (Robot r){
  . . . . . .
}

// Εάν και εφόσον μπορεί, δηλαδή είναι ζωντανό, δεν έχει δραπετεύσει
// ή δεν είναι παγιδευμένο, μετακινείται με τυχαίο τρόπο κατά
// ένα βήμα σε σημείο από το οποίο δεν έχει ξαναπεράσει. Η μέθοδος
// τροποποιεί ανάλογα τα επηρεαζόμενα πεδία και επιστρέφει true εάν
// η μετακίνηση σε κάποιο νέο σημείο ήταν εφικτή, διαφορετικά
// επιστρέφει false. ΝΑ ΟΡΙΣΘΕΙ

public boolean move(){
   . . . . . .
}

// Η παρουσίαση του ρομπότ. Η λειτουργία της μεθόδου διαφαίνεται στο
// πιο κάτω παράδειγμα χρήσης του προγράμματος RandomRobots το οποίο
// είναι client της Robot. ΝΑ ΟΡΙΣΘΕΙ

public String toString(){
   . . . . . .
}

// Βοηθητικές μεθόδους για τη φωτεινότητα (lum) και συμβατότητα
// (compatible) δύο χρωμάτων

private static double lum(Color c) {
      int r = c.getRed();
      int g = c.getGreen();
      int b = c.getBlue();
      return .299*r + .587*g + .114*b;
}

 private static boolean compatible(Color a, Color b) {
   return Math.abs(lum(a) - lum(b)) >= 128.0;
 }
}
```

Το πρόγραμμα RandomRobots.java, η κατασκευή του οποίου **δεν αφορά** την παρούσα ερώτηση, δημιουργεί δεδομένο αριθμό ρομπότ (10 στο πιο κάτω παράδειγμα) σε δεδομένου μεγέθους πλέγμα (20 x 20 στο πιο κάτω παράδειγμα), και αφού τα παρουσιάσει όπως αρχικά εμφανίζονται, διαδοχικά τα κινεί μέχρις ότου κανένα να μην μπορεί να κινηθεί και όταν γίνει αυτό παρουσιάζει την τελική κατάληξη του κάθε ρομπότ. Αρχικά το κάθε ρομπότ τοποθετείται σε διαφορετική θέση. Το πιο κάτω παράδειγμα χρήσης του προγράμματος επιδεικνύει τη λειτουργία της μεθόδου toString():

$java RandomRobots 10 20

```
ROBOTS AT INITIAL STATE

Robot R0 is positioned at (18,7)
The robot is alive

Robot R1 is positioned at (11,14)
The robot is alive

Robot R2 is positioned at (15,19)
The robot is alive

Robot R3 is positioned at (5,0)
The robot is alive and has escaped

Robot R4 is positioned at (17,10)
The robot is alive

Robot R5 is positioned at (8,3)
The robot is alive

Robot R6 is positioned at (12,4)
The robot is alive

Robot R7 is positioned at (14,5)
The robot is alive

Robot R8 is positioned at (10,4)
The robot is alive

Robot R9 is positioned at (0,0)
The robot is alive and has escaped

ROBOTS AT FINAL STATE

Robot R0 is positioned at (17,0)
The robot is alive and has escaped
It has moved as follows:
      - From position (18,7) to position (18,6)
      - From position (18,6) to position (17,6)
      - From position (17,6) to position (16,6)
      - From position (16,6) to position (15,6)
      - From position (15,6) to position (15,5)
      - From position (15,5) to position (15,4)
      - From position (15,4) to position (14,4)
      - From position (14,4) to position (14,3)
      - From position (14,3) to position (14,2)
      - From position (14,2) to position (14,1)
      - From position (14,1) to position (15,1)
      - From position (15,1) to position (15,2)
      - From position (15,2) to position (16,2)
      - From position (16,2) to position (17,2)
      - From position (17,2) to position (17,1)
      - From position (17,1) to position (17,0)

Robot R1 is positioned at (5,15)
The robot is alive but is trapped
It has moved as follows:
      - From position (11,14) to position (10,14)
      - From position (10,14) to position (10,13)
```

```
      - From position (10,13) to position (11,13)
      - From position (11,13) to position (11,12)
      - From position (11,12) to position (10,12)
      - From position (10,12) to position (9,12)
      - From position (9,12) to position (8,12)
      - From position (8,12) to position (8,13)
      - From position (8,13) to position (9,13)
      - From position (9,13) to position (9,14)
      - From position (9,14) to position (8,14)
      - From position (8,14) to position (8,15)
      - From position (8,15) to position (7,15)
      - From position (7,15) to position (7,14)
      - From position (7,14) to position (6,14)
      - From position (6,14) to position (6,15)
      - From position (6,15) to position (6,16)
      - From position (6,16) to position (5,16)
      - From position (5,16) to position (4,16)
      - From position (4,16) to position (4,15)
      - From position (4,15) to position (4,14)
      - From position (4,14) to position (5,14)
      - From position (5,14) to position (5,15)

Robot R2 is positioned at (15,20)
The robot is alive and has escaped
It has moved as follows:
      - From position (15,19) to position (15,20)

Robot R3 is positioned at (5,0)
The robot is alive and has escaped

Robot R4 is positioned at (14,0)
The robot is alive and has escaped
It has moved as follows:
      - From position (17,10) to position (16,10)
      - From position (16,10) to position (16,9)
      - From position (16,9) to position (17,9)
      - From position (17,9) to position (17,8)
      - From position (17,8) to position (16,8)
      - From position (16,8) to position (15,8)
      - From position (15,8) to position (15,7)
      - From position (15,7) to position (15,6)
      - From position (15,6) to position (14,6)
      - From position (14,6) to position (14,5)
      - From position (14,5) to position (14,4)
      - From position (14,4) to position (13,4)
      - From position (13,4) to position (13,3)
      - From position (13,3) to position (13,2)
      - From position (13,2) to position (12,2)
      - From position (12,2) to position (12,1)
      - From position (12,1) to position (13,1)
      - From position (13,1) to position (14,1)
      - From position (14,1) to position (14,0)

Robot R5 is positioned at (8,0)
The robot is alive and has escaped
It has moved as follows:
      - From position (8,3) to position (7,3)
      - From position (7,3) to position (7,2)
      - From position (7,2) to position (8,2)
      - From position (8,2) to position (8,1)
      - From position (8,1) to position (8,0)
```

```
Robot R6 is positioned at (10,7)
The robot is dead
It has moved as follows:
      - From position (12,4) to position (12,5)
      - From position (12,5) to position (12,6)
      - From position (12,6) to position (12,7)
      - From position (12,7) to position (11,7)
      - From position (11,7) to position (11,8)
      - From position (11,8) to position (10,8)
      - From position (10,8) to position (10,7)

Robot R7 is positioned at (11,0)
The robot is alive and has escaped
It has moved as follows:
      - From position (14,5) to position (13,5)
      - From position (13,5) to position (12,5)
      - From position (12,5) to position (12,4)
      - From position (12,4) to position (12,3)
      - From position (12,3) to position (11,3)
      - From position (11,3) to position (11,2)
      - From position (11,2) to position (11,1)
      - From position (11,1) to position (11,0)

Robot R8 is positioned at (10,7)
The robot is dead
It has moved as follows:
      - From position (10,4) to position (11,4)
      - From position (11,4) to position (12,4)
      - From position (12,4) to position (12,5)
      - From position (12,5) to position (11,5)
      - From position (11,5) to position (11,6)
      - From position (11,6) to position (10,6)
      - From position (10,6) to position (10,7)

Robot R9 is positioned at (0,0)
The robot is alive and has escaped
```

ΤΕΛΟΣ ΕΡΩΤΗΣΕΩΝ

ΧΩΡΟΣ ΑΠΑΝΤΗΣΕΩΝ

Ονοματεπώνυμο Φοιτητή: ----------

Ταυτότητα: ----------

Υπογραφή: ----------

Ερωτήσεις	**Μονάδες**
Ερώτηση 1	
Ερώτηση 2	
Ερώτηση 3	
Σύνολο Μονάδων	

Παρατηρήσεις Διδάσκοντα

Απάντηση στην Ερώτηση 1

(τυχόν συνέχεια στην απάντηση της Ερώτησης 1)

(τυχόν συνέχεια στην απάντηση της Ερώτησης 1)

Απάντηση στην Ερώτηση 2

(τυχόν συνέχεια στην απάντηση της Ερώτησης 2)

(τυχόν συνέχεια στην απάντηση της Ερώτησης 2)

Απάντηση στην Ερώτηση 3

(τυχόν συνέχεια στην απάντηση της Ερώτησης 3)

(τυχόν συνέχεια στην απάντηση της Ερώτησης 3)

Επιπρόσθετο φύλλο (για συμπλήρωση απάντησης ή πρόχειρο)

Επιπρόσθετο φύλλο (για συμπλήρωση απάντησης ή πρόχειρο)

Επιπρόσθετο φύλλο (για συμπλήρωση απάντησης ή πρόχειρο)

Επιπρόσθετο φύλλο (για συμπλήρωση απάντησης ή πρόχειρο)

ΤΕΛΟΣ ΕΞΕΤΑΣΤΙΚΟΥ ΔΟΚΙΜΙΟΥ