Δεύτερη Σειρά ασκήσεων
Ημερομηνία Παράδοσης: Δευτέρα 8 Ιουνίου 3:00 μ.μ.

Για την άσκηση αυτή θα προγραμματίσετε σε Java μια προσομοίωση της εξέλιξης ενός πληθυσμού κυνηγών και θηραμάτων. Σε αυτή την προσομοίωση τα θηράματα είναι κουνέλια και οι κυνηγοί αλεπούδες. Τα ζώα είναι περιορισμένα μέσα σε ένα δισδιάστατο πλέγμα κελιών 20×20. Μόνο ένα ζώο μπορεί να είναι σε ένα κελί ανα πάσα στιγμή. Τα ζώα μετακινούνται μόνο μεταξύ γειτονικών κελιών (πάνω, κάτω, δεξιά, αριστερά) και δεν μπορούν να βγουν εκτός του πλέγματος. Ο χρόνος της προσομοίωσης κυλάει σε διακριτά βήματα. Κάθε ζώο κάνει κάποιες δράσεις σε κάθε βήμα.

Συγκεκριμένα, τα κουνέλια κάνουν τις εξής δράσεις:

- Κίνηση: Σε κάθε βήμα ένα κουνέλι επιλέγει τυχαία ένα από τα γειτονικά κελιά. Αν το κελί είναι εντός του πλέγματος και δεν είναι κατειλημμένο μετακινείται σε αυτό το κελί. Αλλιώς μένει στο κελί στο οποίο βρίσκεται.
- Αναπαραγωγή: Αν το κουνέλι επιβιώσει για τρία βήματα, στο τέλος του τρίτου βήματος προσπαθεί να αναπαραχθεί. Αυτό προσομοιώνεται δημιουργώντας ένα καινούριο κουνέλι σε ένα γειτονικό κελί το οποίο είναι άδειο. Αν δεν υπάρχει άδειο γειτονικό κελί, τότε το κουνέλι δεν αναπαράγεται. Αν περάσει το βήμα της αναπαραγωγής (είτε επιτυχώς, είτε ανεπιτυχώς), η επόμενη προσπάθεια είναι μετά από άλλα τρία βήματα.

Οι αλεπούδες κάνουν τις εξής δράσεις:

- Κίνηση: Σε κάθε βήμα, αν υπάρχει ένα γειτονικό κελί το οποίο να περιέχει ένα κουνέλι, η αλεπού μετακινείται σε αυτό το κελί και τρώει το κουνέλι. Αλλιώς μετακινείται με τον ίδιο τρόπο όπως το κουνέλι. Η αλεπού δεν μπορεί να φάει άλλη αλεπού.
- Αναπαραγωγή: Η αναπαραγωγή γίνεται με τον ίδιο ακριβώς τρόπο όπως και με τα κουνέλια, αλλά κάθε οκτώ βήματα αντί για κάθε τρία.
- Θάνατος από ασιτεία: Αν η αλεπού δεν φάει κουνέλι για τρία βήματα τότε πεθαίνει από ασιτεία.

Στην προσομοίωση θα δρουν πρώτα οι αλεπούδες και μετά τα κουνέλια.

Θα φτιάξετε ένα πρόγραμμα που υλοποιεί την παραπάνω προσομοίωση. Για την υλοποίηση θα δημιουργήσετε τις εξής κλάσεις:

- **Cell**: Μια κλάση που κρατάει πληροφορίες για ένα κελί του πλέγματος.
- **Grid**: Μια κλάση που δημιουργεί το πλέγμα και υλοποιεί τις βασικές του λειτουργίες
- **Animal**: Αφηρημένη κλάση που κρατάει γενική πληροφορία για ένα ζώο (αλεπού ή κουνέλι).
- **Rabbit**: Ενυπόστατη κλάση που κληρονομεί από την κλάση Animal και υλοποιεί την συμπεριφορά του κουνελιού.
- **Fox**: Ενυπόστατη κλάση που κληρονομεί από την κλάση Animal και υλοποιεί την συμπεριφορά της αλεπούς.
- **PopulationSimulator**: Μια κλάση που υλοποιεί την προσομοίωση.
- **Simulation**: Μια κλάση με την main που απλά τρέχει τον προσομοιωτή.

Οι κλάσεις αλληλοεπιδρούν μεταξύ τους, οπότε για να κάνετε compile μια κλάση θα πρέπει να δημιουργήσετε και αυτές που χρησιμοποιεί, έστω και με κενό κώδικα.

Θα περιγράψουμε τώρα με λεπτομέρεια την κάθε κλάση:

Cell: Η κλάση αυτή κρατάει πληροφορίες για ένα κελί του πλέγματος. Το κάθε κελί κρατάει μια λίστα (ArrayList) με τα γειτονικά κελιά (neighbors), και το ζώο (Animal) το οποίο βρίσκεται μέσα στο κελί (animal). Αν δεν υπάρχει κάποιο ζώο αυτό είναι null. Η κλάση έχει τις εξής μεθόδους:

- Μέθοδο πρόσβασης (**accessor** method) για το πεδίο neighbors με τη λίστα των γειτόνων.
- **addNeighbor**: Παίρνει όρισμα ένα κελί και το προσθέτει στη λίστα των γειτόνων.
- **getRandomNeighbor**: Επιστρέφει ένα τυχαίο γειτονικό κελί. Η υλοποίηση θα πρέπει να προβλέπει και την περίπτωση που είμαστε στα άκρα του πλέγματος και το γειτονικό κελί είναι εκτός του πλέγματος, και στην περίπτωση αυτή θα επιστρέφει null. Στην περίπτωση αυτή τα γειτονικά κελιά είναι λιγότερα από τέσσερα. Δημιουργήστε ένα τυχαίο αριθμό μεταξύ 0 και 3, και επιστρέψετε την αντίστοιχη θέση στη λίστα αν είσαστε μέσα στα όρια της λίστας, ή null αν είσαστε εκτός.
- Μεθόδους πρόσβασης και μετάλλαξης (**accessor** και **mutator** μεθόδους) για το πεδίο animal.
- **removeAnimal**: Αφαιρεί το ζώο από το κελί, δηλαδή θέτει το πεδίο στην τιμή null.
- **isEmpty**: Επιστρέφει Boolean τιμή αν το κελί είναι ελεύθερο ή όχι (αν έχει ζώο ή όχι).
- **containsRabbit**: Επιστρέφει Boolean τιμή αν το κελί έχει ζώο και αυτό είναι κουνέλι. Για την μέθοδο αυτή θα χρειαστείτε την μέθοδο isRabbit της κλάσης Animal.

Αποθηκεύστε τον κώδικα σας στο αρχείο **Cell.java**

Υποδείξεις:

- Αν θέλετε να κάνετε compile την κλάση Cell μόνη της, ορίστε την αφηρημένη κλάση Animal μόνο με τις αφηρημένες μεθόδους.
- Το κάθε κελί έχει κάποιες συντεταγμένες στο πλέγμα. Δεν σας χρειάζονται στην υλοποίηση, αλλά ενδέχεται να σας χρειαστούν στο debugging. Στην περίπτωση αυτή θα χρειαστεί να προσθέσετε επιπλέον πεδία για τις συντεταγμένες, αντίστοιχο constructor, και μια toString που να τις επιστρέφει.

Grid: Η κλάση αυτή κρατάει πληροφορίες για το πλέγμα. Έχει ένα πεδίο SIZE το οποίο έχει την τιμή 20 και είναι σταθερά (βάλετε την λέξη final πριν από το int). Επίσης έχει ένα πίνακα SIZE × SIZE με κελιά. Η κλάση έχει τις εξής μεθόδους:

- Τον **constructor** ο οποίος δημιουργεί τα κελιά, τα τοποθετεί στον πίνακα και τα συνδέει μεταξύ τους. Για την σύνδεση των κελιών αρκεί να προσθέσετε το κελί (i,j) στους γείτονες του κελιού (i-1,j) (αν i>0) και (i,j-1) (αν j>0) και αντίστοιχα αυτά τα κελιά στους γείτονες του (i,j).
- **addRandomly**: Παίρνει όρισμα ένα Animal και το προσθέτει σε ένα τυχαία επιλεγμένο άδειο κελί. Ενημερώνει και το κελί του αντικειμένου Animal .
- **toString**: Επιστρέφει ένα String με το πλέγμα. Αν ένα κελί είναι άδειο θα προσθέτει το String "_". Αν υπάρχει ζώο θα προστίθεται η String αναπαράσταση του ζώου στο κελί. Δείτε παραδείγματα της String αναπαράστασης στο τέλος της εκφώνησης.

Αποθηκεύστε τον κώδικα σας στο αρχείο **Grid.java**

Υπόδειξη: Για να τεστάρετε την κλάση αλλάξετε την toString ώστε τοποθετεί "_" για τα άδεια κελιά και "*" για τα γεμάτα. Επίσης, εκτυπώστε τους γείτονες διαφορετικών κελιών για να βεβαιωθείτε ότι τα κελιά συνδέονται σωστά.

Animal: Η κλάση Animal είναι αφηρημένη κλάση που κρατάει πληροφορίες για ένα ζώο (είτε κουνέλι, είτε αλεπού). Έχει πεδία το κελί (Cell) στο οποίο βρίσκεται το ζώο και τον αριθμό των βημάτων που έχει επιβιώσει το ζώο.
Έχει τις εξής *αφηρημένες* μεθόδους:

- **isRabbit**: Επιστρέφει true ή false αν το ζώο είναι κουνέλι ή όχι.
- **timeToBreed**: Επιστρέφει true ή false αν σε αυτό το βήμα μπορεί να αναπαραχθεί το ζώο (πολλαπλάσιο του 3 για το κουνέλι, πολλαπλάσιο του 8 για την αλεπού).
- **giveBirth**: Επιστρέφει ένα αντικείμενο Animal το οποίο γεννάει το ζώο.
- **move**: Υλοποιεί την κίνηση του ζώου.

Οι αφηρημένες αυτές μέθοδοι θα υλοποιηθούν από τις παράγωγες κλάσεις.

Η κλάση έχει επίσης τις εξής ενυπόστατες μεθόδους:

- Μεθόδους πρόσβασης (**accessor**) και μετάλλαξης (**mutator**) όπου χρειάζεται.
- **survived**: Αυξάνει τον αριθμό των βημάτων που επιβίωσε το ζώο.
- **randomMove**: Επιλέγει ένα τυχαίο γειτονικό κελί και αν είναι μέσα στα όρια και είναι άδειο, μετακινεί το ζώο. Η μετακίνηση θα πρέπει να ενημερώσει και τα πεδία του ζώου, αλλά και τα κελιά από και προς τα οποία γίνεται η μετακίνηση.
- **breed**: Η μέθοδος αυτή υλοποιεί την αναπαραγωγή. Ελέγχει αν είναι βήμα αναπαραγωγής, και αν υπάρχει άδειο γειτονικό κελί. Αν ναι, καλεί την giveBirth, τοποθετεί το ζώο στο κελί και ενημερώνει το κελί του ζώου. Επιστρέφει το αντικείμενο Animal με το νεογέννητο ζώο, ή null αν δεν έγινε αναπαραγωγή.

Αποθηκεύστε τον κώδικα σας στο αρχείο **Animal.java**

Rabbit: Η κλάση Rabbit κληρονομεί από την κλάση Animal και κρατάει πληροφορίες για ένα κουνέλι. Υλοποιεί όλες τις αφηρημένες μεθόδους όπως ορίζει η συμπεριφορά του κουνελιού. Η move καλεί την survived για να ενημερώσει τα βήματα επιβίωσης του κουνελιού. Η κλάση έχει επιπλέον:

- Ένα boolean πεδίο eaten που μας λέει αν το κουνέλι φαγώθηκε, και μεθόδους **becomeEaten**, και **isEaten** που θέτουν και επιστρέφουν το πεδίο αντίστοιχα.
- Την μέθοδο **toString** η οποία επιστρέφει το String “o”.

Αποθηκεύστε τον κώδικα σας στο αρχείο **Rabbit.java**

Fox: Η κλάση Fox κληρονομεί από την κλάση Animal και κρατάει πληροφορίες για μία αλεπού. Υλοποιεί όλες τις αφηρημένες μεθόδους όπως ορίζει η συμπεριφορά της αλεπούς. Η μέθοδος move κοιτάει τα γειτονικά κελιά και αν υπάρχει κάποιο στο οποίο υπάρχει κουνέλι μετακινείται σε αυτό και το τρώει. Αν δεν υπάρχει, κάνει μια τυχαία κίνηση. Η μέθοδος καλεί την survived για να ενημερώσει τα βήματα επιβίωσης της αλεπούς. Επιπλέον:

- Η κλάση χρειάζεται να ξέρει πόσα βήματα έχουν περάσει από την τελευταία φορά που έφαγε η αλεπού και να ενημερώνει το πεδίο κατάλληλα στην move.
- Η κλάση έχει την μέθοδο **starve** η οποία ελέγχει αν έχουν περάσει τρία βήματα από την τελευταία φορά που έφαγε η αλεπού, και αν ναι την αφαιρεί από το κελί της. Η μέθοδος επιστρέφει true ή false ανάλογα.
- Η κλάση έχει την μέθοδο **toString** η οποία επιστρέφει το String “X”.

Αποθηκεύστε τον κώδικα σας στο αρχείο **Fox.java**

PopulationSimulator: Η κλάση αυτή κρατάει την πληροφορία που χρειάζεται για να κάνει το simulation. Έχει πεδία ένα πλέγμα (Grid), και δύο HashSet τα οποία κρατάνε τα κουνέλια και τις αλεπούδες που έχουμε στην προσομοίωση. Επίσης δύο σταθερές NUM_OF_FOXES και NUM_OF_RABBITS που είναι ο αρχικός αριθμός των αλεπούδων και κουνελιών αντίστοιχα. Στην υλοποίηση σας θα τα θέσετε στις τιμές 5 και 100 αντίστοιχα. Η κλάση έχει επίσης τις εξής μεθόδους:

- **populateGrid**: Βοηθητική private μέθοδος η οποία προσθέτει τα ζώα στο πλέγμα σε τυχαίες θέσεις. Κάθε ζώο που προστίθεται στο πλέγμα, προστίθεται και στο αντίστοιχο HashSet.
- **handleFoxes**: Βοηθητική μέθοδος η οποία χειρίζεται τις δράσεις των αλεπούδων. Για κάθε αλεπού στο HashSet κάνουμε κίνηση, ελέγχουμε για θάνατο από ασιτεία, και αν έχει επιβιώσει γίνεται αναπαραγωγή αν είναι βήμα αναπαραγωγής. Οι αλεπούδες που πεθαίνουν πρέπει να αφαιρεθούν από το HashSet ενώ αυτές που γεννιούνται πρέπει να προστεθούν. (Προσοχή στην αφαίρεση και πρόσθεση στο HashSet. Δείτε την υπόδειξη παρακάτω).
- **handleRabbits**: Βοηθητική μέθοδος η οποία χειρίζεται τις δράσεις των κουνελιών. Για κάθε κουνέλι ελέγχουμε αν έχει φαγωθεί, αν όχι, κάνουμε την κίνηση και μετά αναπαραγωγή, αν είναι βήμα αναπαραγωγής. Τα κουνέλια που φαγώθηκαν πρέπει να αφαιρεθούν από το HashSet, ενώ αυτά που γεννιούνται πρέπει να προστεθούν. (Προσοχή στην αφαίρεση και πρόσθεση στο HashSet. Δείτε την υπόδειξη παρακάτω).

- **simulate**: Υλοποιεί την προσομοίωση. Παίρνει σαν όρισμα τον αριθμό των βημάτων. Αρχικά προσθέτει τα ζώα στο πλέγμα και μετά υλοποιεί τα βήματα της προσομοίωσης. Σε κάθε βήμα πρώτα χειρίζεται τις αλεπούδες και μετά τα κουνέλια. Στην αρχή του βήματος τυπώνει το πλέγμα και περιμένει από τον χρήστη να πατήσει enter για να συνεχίσει.

Αποθηκεύστε τον κώδικα σας στο αρχείο **PopulationSimulator.java**

Υπόδειξη: Δεν μπορείτε να προσθέσετε ή να αφαιρέσετε στοιχεία από ένα HashSet ενώ το διατρέχετε. Αποθηκεύστε τα στοιχεία που θέλετε να προσθέσετε και να αφαιρέσετε σε ξεχωριστές λίστες. Προσθέστε και αφαιρέστε τα στοιχεία αφού τελειώσετε με την διάσχιση του HashSet.

Simulation: Η κλάση αυτή έχει την main. Δημιουργεί το αντικείμενο PopulationSimulator, ζητάει από τον χρήστη να προσδιορίσει τον αριθμό των βημάτων και καλεί την simulate.

Αποθηκεύστε τον κώδικα σας στο αρχείο **Simuation.java**

Στο Παράρτημα στο τέλος της εκφώνησης σας δίνεται ένα παράδειγμα της εκτέλεσης της προσομοίωσης για να δείτε την String αναπαράσταση του πλέγματος. Εκτός από το πλέγμα τυπώνεται και ο αριθμός των κουνελιών και αλεπούδων. Δεν είναι απαραίτητο να έχετε αυτή την εκτύπωση, αλλά εκτός του πλέγματος μπορείτε να τυπώσετε και άλλη πληροφορία που είναι χρήσιμη για να παρακολουθήσετε την προσομοίωση, αρκεί να μην χάνεται η εκτύπωση του πλέγματος.

Στην προσομοίωση θα δείτε μια κυκλική συμπεριφορά, όπου θα εναλλάσσονται τα κουνέλια και οι αλεπούδες ως κυρίαρχα στο πλέγμα.

ΓΕΝΙΚΕΣ ΟΔΗΓΙΕΣ

- Μια κλάση που δεν κάνει compile μηδενίζεται αυτόματα.
- Δεν επιτρέπεται η χρήση public ή protected πεδίων στην άσκηση. Επίσης ο κώδικας θα πρέπει να είναι σωστά στοιχισμένος και καλά γραμμένος. Θα αφαιρεθούν βαθμοί από προγράμματα που είναι πολύ κακά γραμμένα.
- Θα τεστάρουμε και θα βαθμολογήσουμε την κάθε κλάση ξεχωριστά. Γι αυτό και θα πρέπει να σώσετε την κάθε κλάση σε ξεχωριστό αρχείο. Θα πρέπει επίσης να κρατήσετε τα ονόματα και τα ορίσματα των public μεθόδων ακριβώς όπως σας ζητούνται.
- Κάντε turnin τα προγράμματα σας στο assignment2@myy205.

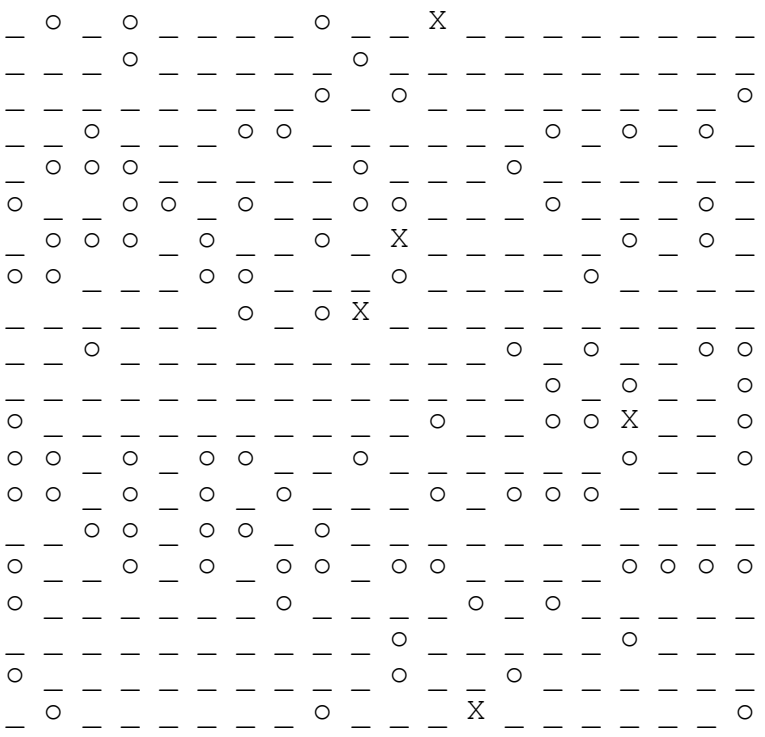
  π.χ. turnin assignment2@myy205 Simulation.java

  Μπορείτε να κάνετε turnin πολλά αρχεία μαζί στην ίδια εντολή. Διαβάστε προσεκτικά τις οδηγίες για το trunin στο ecourse και βεβαιωθείτε ότι μπορείτε να κάνετε την διαδικασία κάποιες μέρες πριν την προθεσμία. Μπορειτε να κάνετε πολλαπλές φορές turnin το ίδιο αρχείο.

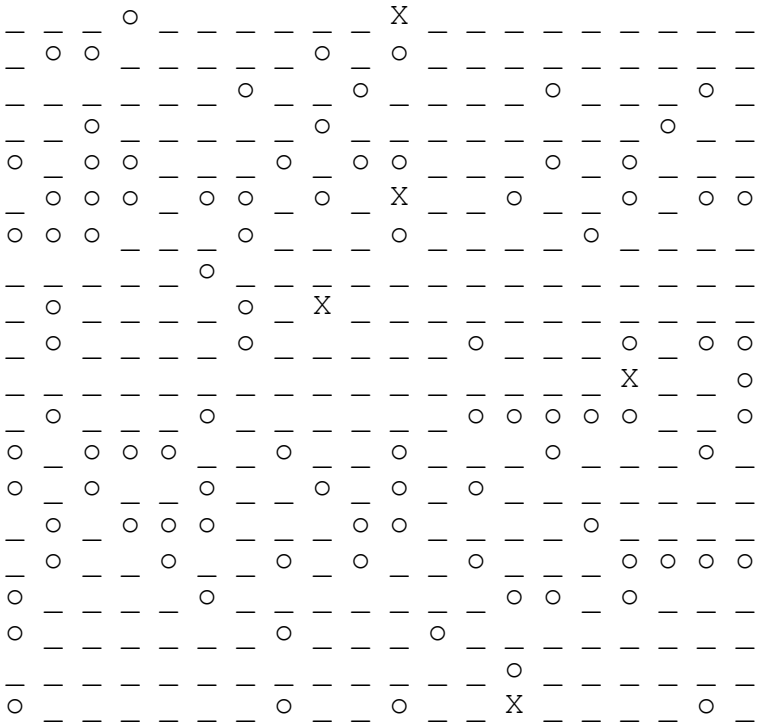
  Στον κώδικα να αναγράφονται σε σχόλια το όνομα και ο ΑΜ σας (όχι με λατινικούς χαρακτήρες).

Παράρτημα

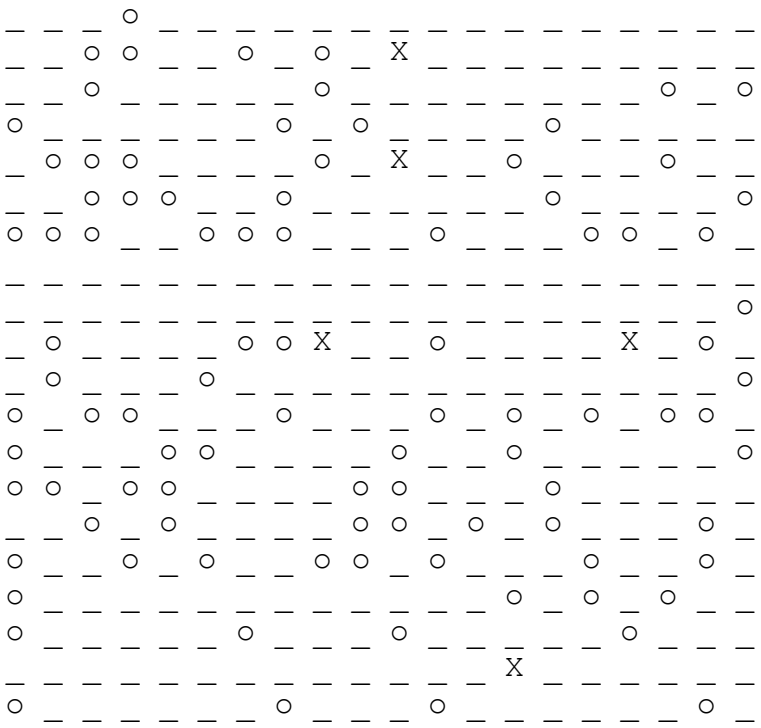
```
C:\[…]\assignment2>java Simulation
Give the number of steps:
1000
```



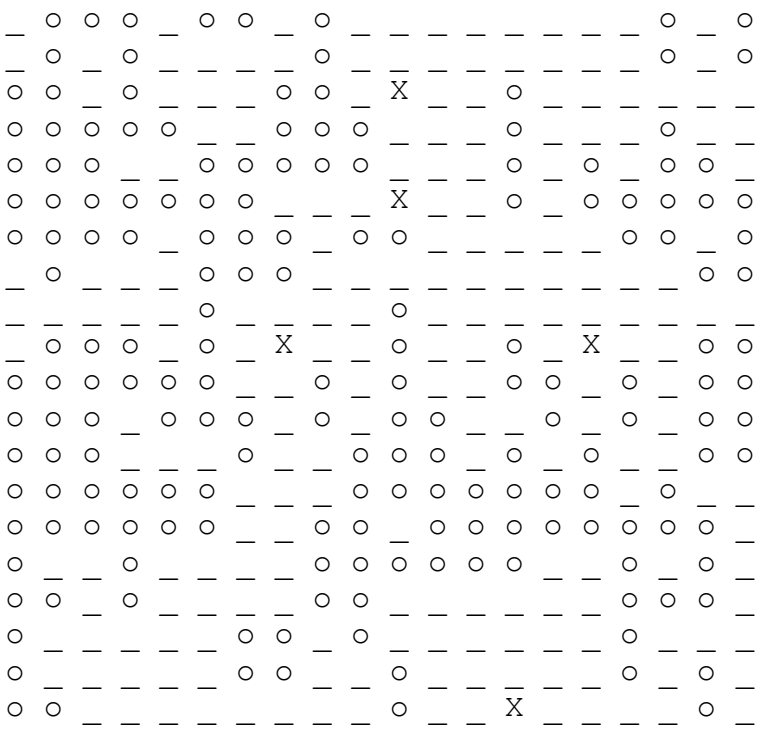
```
100 rabbits and 5 foxes
```



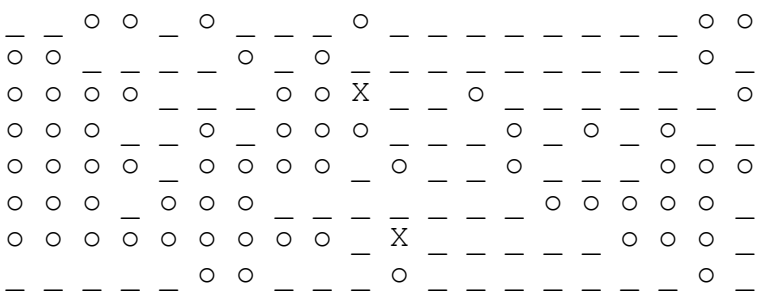
```
97 rabbits and 5 foxes
```

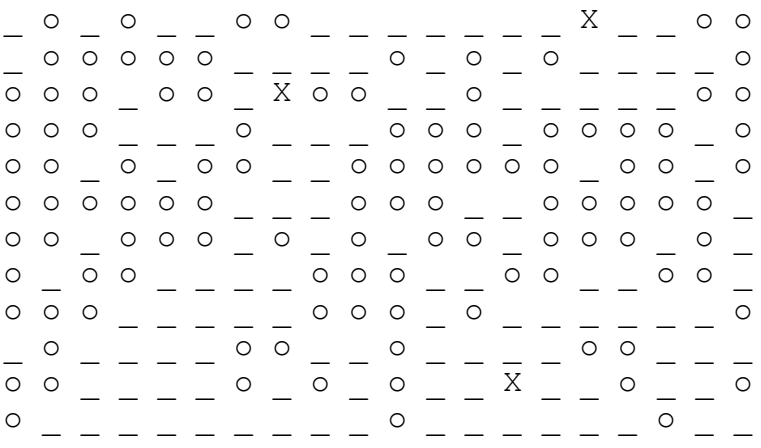


93 rabbits and 5 foxes

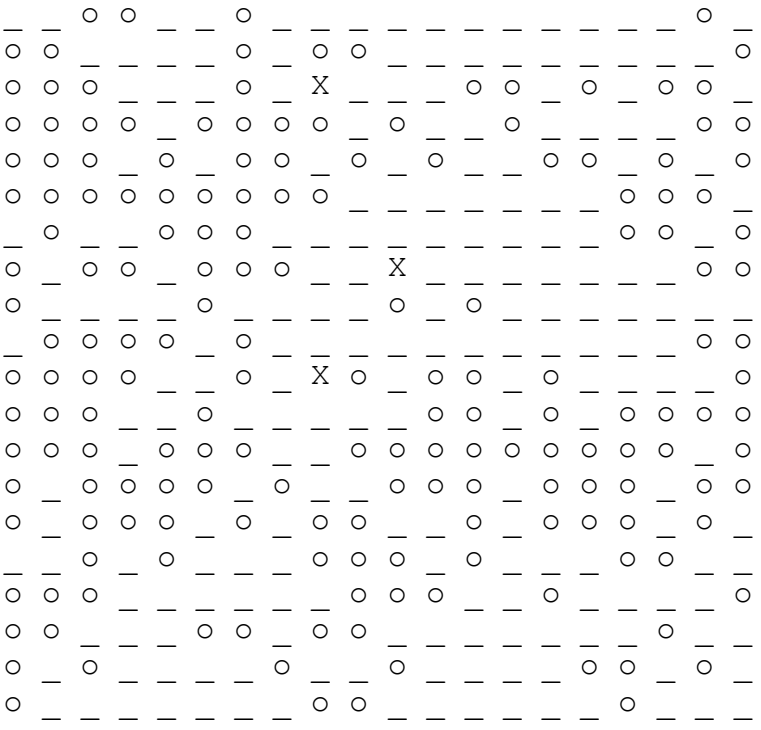


182 rabbits and 5 foxes

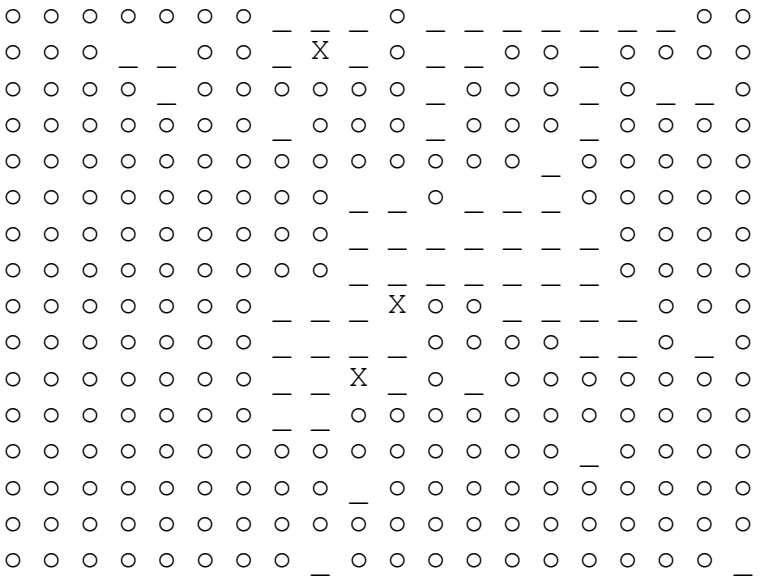


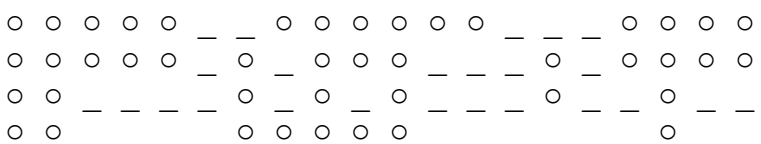


181 rabbits and 5 foxes

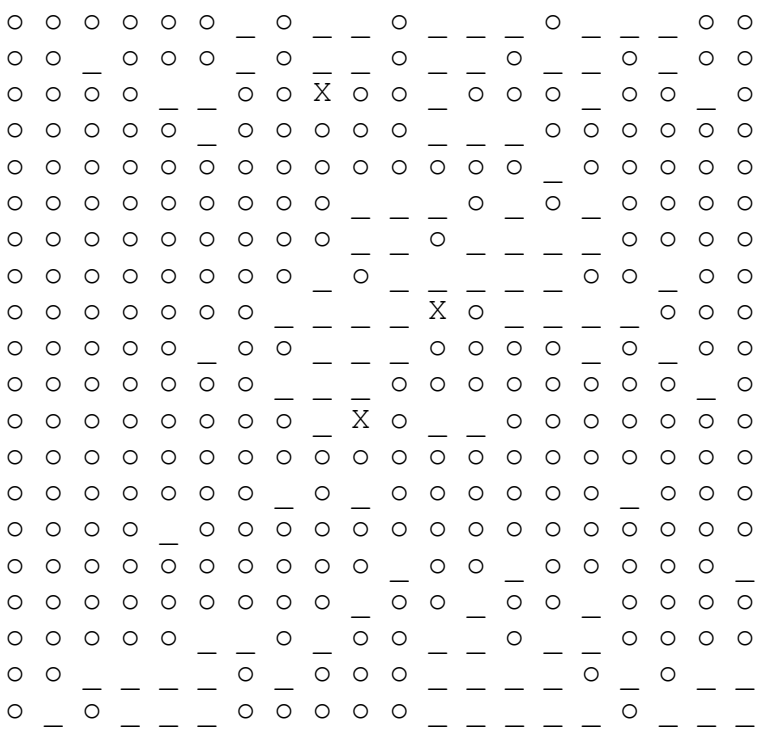


178 rabbits and 3 foxes

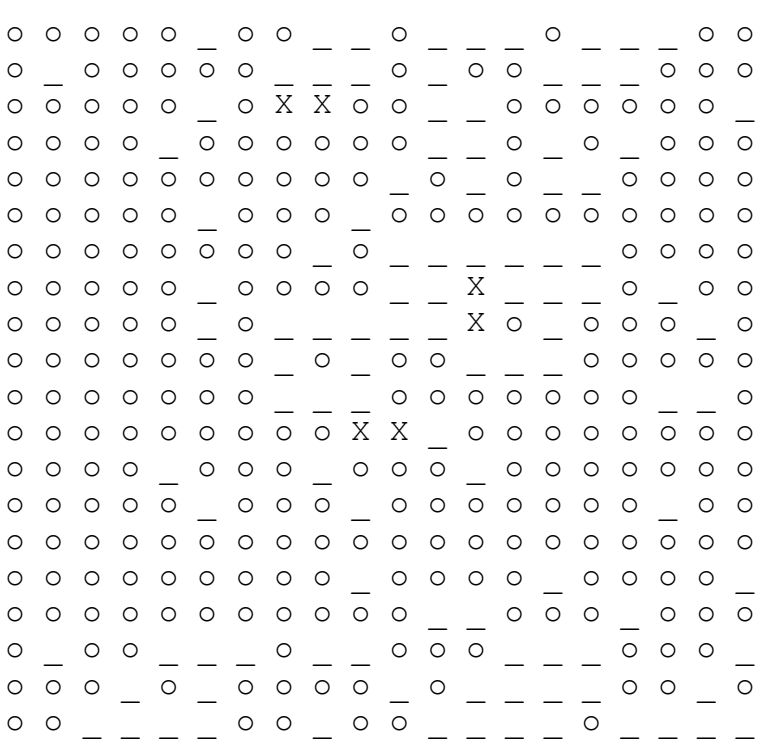




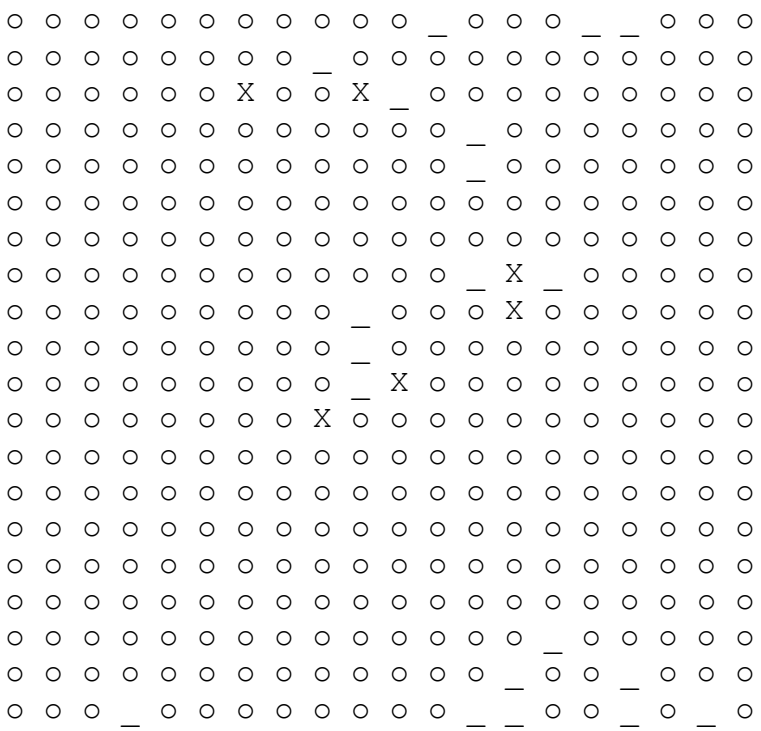
292 rabbits and 3 foxes



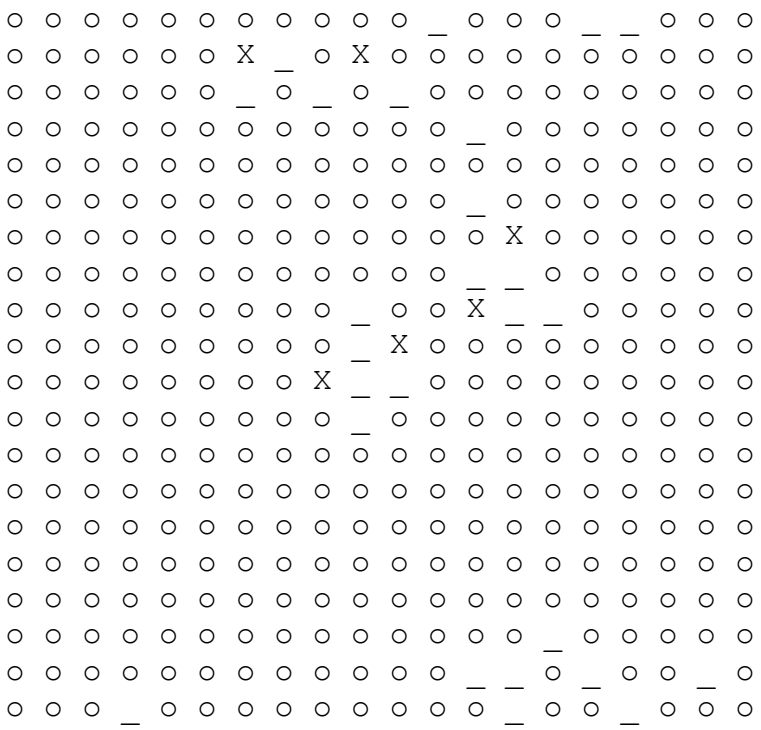
289 rabbits and 3 foxes



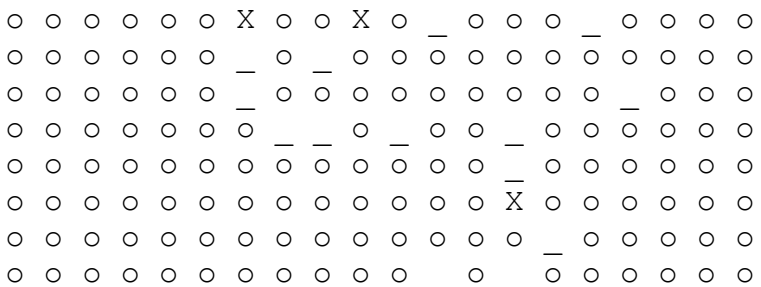
286 rabbits and 6 foxes



374 rabbits and 6 foxes



368 rabbits and 6 foxes



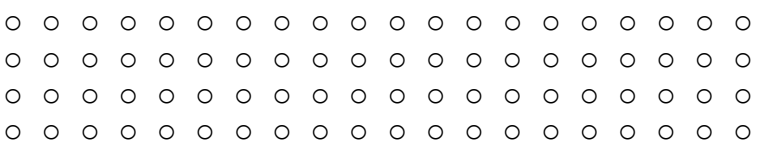
```
o o o o o o o o o _ X X _ o o o o o o o o
o o o o o o o o o X _ o _ o _ _ o o o o o
o o o o o o o o _ o _ o o o o o o o o o o
o o o o o o o o o o _ _ o o o o o o o o o
o o o o o o o o o o o o o o o o o o o o o
o o o o o o o o o o o o o o o o o o o o o
o o o o o o o o o o o o o o o o o o o o o
o o o o o o o o o o o o o o o o o o o o o
o o o o o o o o o o o o o o o o o o o _ o
o o o o o o o o o o o o o o _ o o o o o o
o o o o o o o o o o o o o _ _ o o _ o o o
o o _ o o o o o o o o o _ o o o _ o o o
```

362 rabbits and 6 foxes

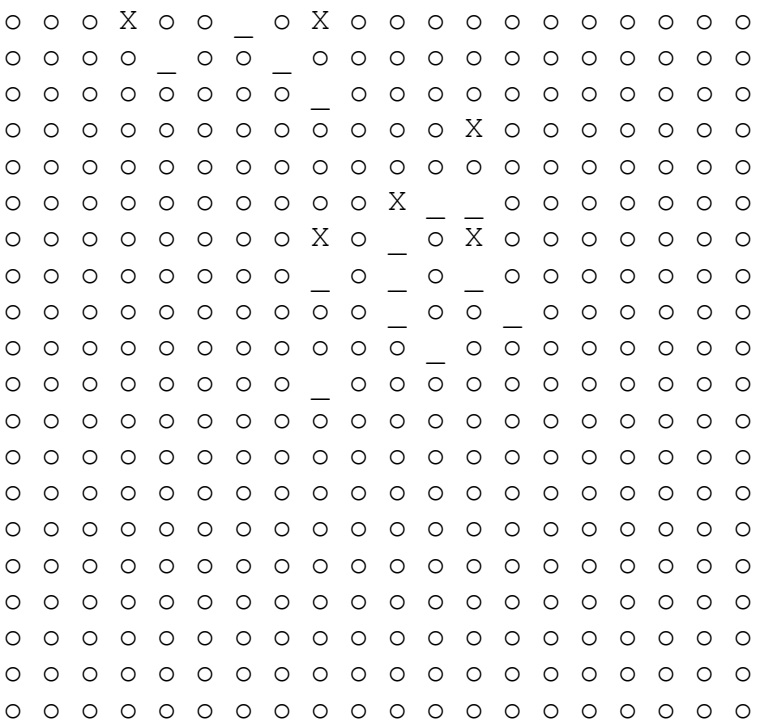
```
o o o o o X o o X o o o o o o o o o o o
o o o o o o o o o o o o o o o o o o o o
o o o o o o o o o o o o o o o o o o o o
o o o o o o o o o o o o o o o o o o o o
o o o o o o o o o o o o o o o o o o o o
o o o o o o o o o o o o X o o o o o o o
o o o o o o o o o o o o o o o o o o o o
o o o o o o o o o o X o o o o o o o o o
o o o o o o o o o X o o _ X o o o o o o
o o o o o o o o o o o o o o o o o o o o
o o o o o o o o o o _ o o o o o o o o o
o o o o o o o o o o o o o o o o o o o o
o o o o o o o o o o o o o o o o o o o o
o o o o o o o o o o o o o o o o o o o o
o o o o o o o o o o o o o o o o o o o o
o o o o o o o o o o o o o o o o o o o o
o o o o o o o o o o o o o o o o o o o o
o o o o o o o o o o o o o o o o o o o o
o o o o o o o o o o o o o o o o o o o o
```

392 rabbits and 6 foxes

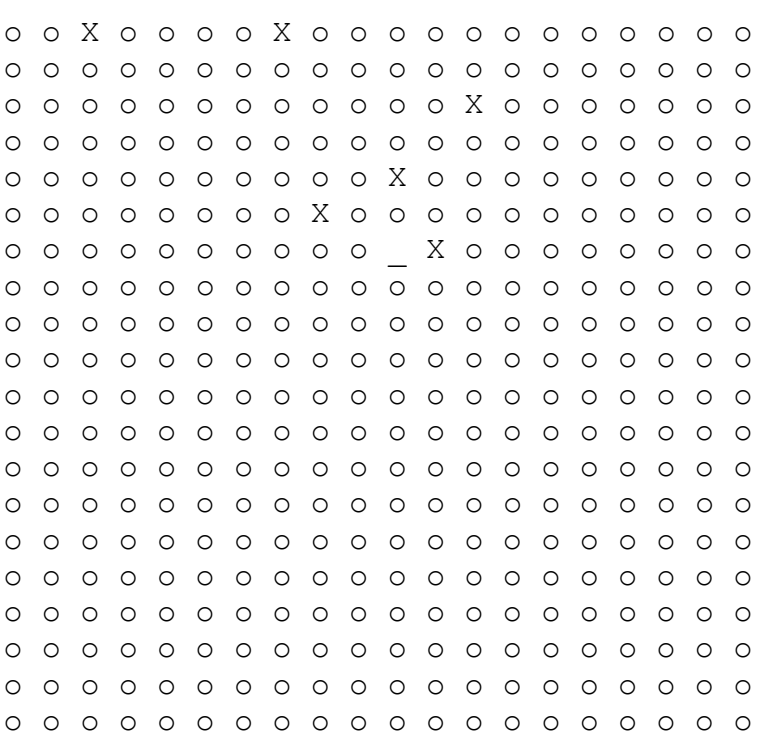
```
o o o o X o _ X o o o o o o o o o o o o
o o o o o o o o o o o o o o o o o o o o
o o o o o o o o _ o o o o o o o o o o o
o o o o o o o o o o o o o o o o o o o o
o o o o o o o o o o o o o X o o o o o o
o o o o o o o o o o o _ o o o o o o o o
o o o o o o o o o o X o o o o o o o o o
o o o o o o o o X o _ o X o o o o o o o
o o o o o o o o _ o o _ o _ o o o o o o
o o o o o o o o o o o o o o o o o o o o
o o o o o o o o o _ o o o o o o o o o o
o o o o o o o o o o o o o o o o o o o o
o o o o o o o o o o o o o o o o o o o o
o o o o o o o o o o o o o o o o o o o o
o o o o o o o o o o o o o o o o o o o o
```



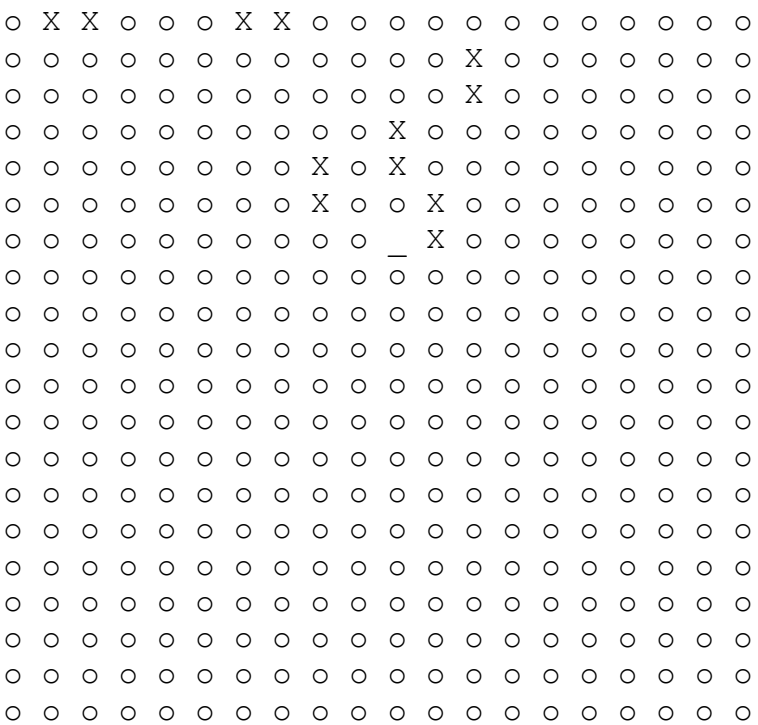
386 rabbits and 6 foxes



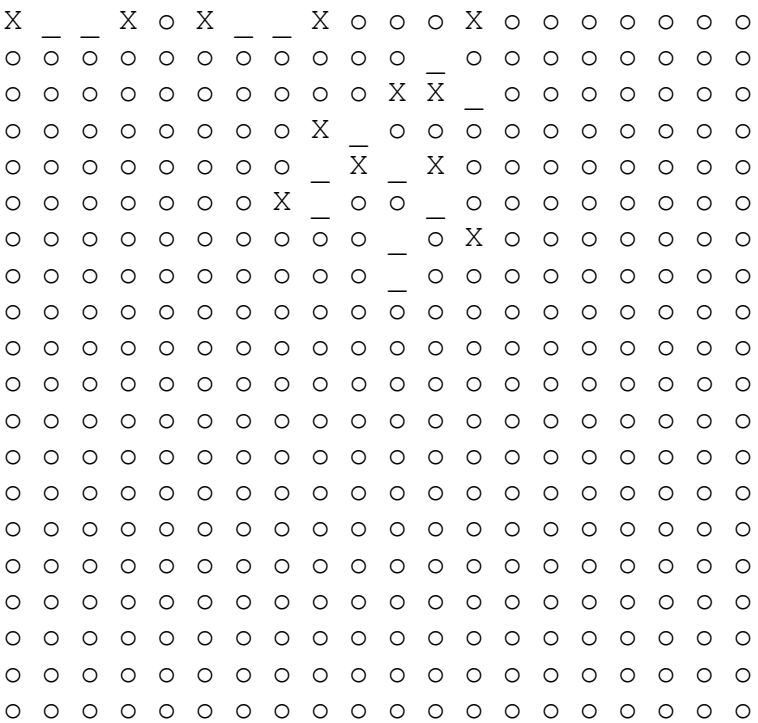
380 rabbits and 6 foxes



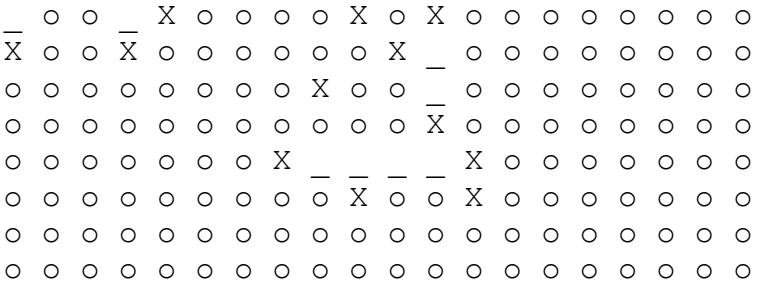
393 rabbits and 6 foxes

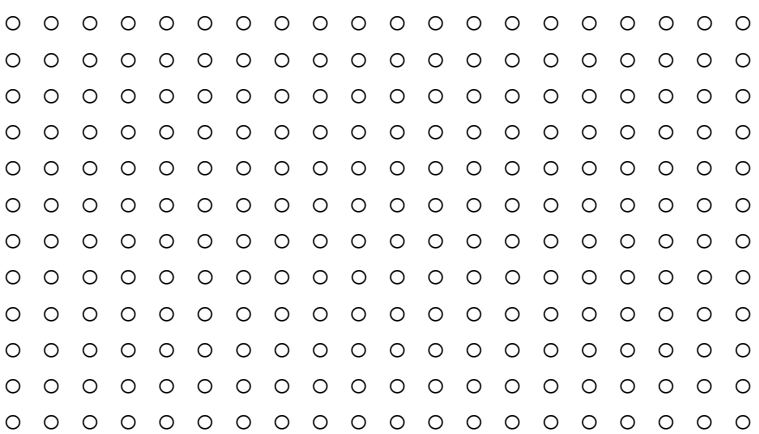


387 rabbits and 12 foxes

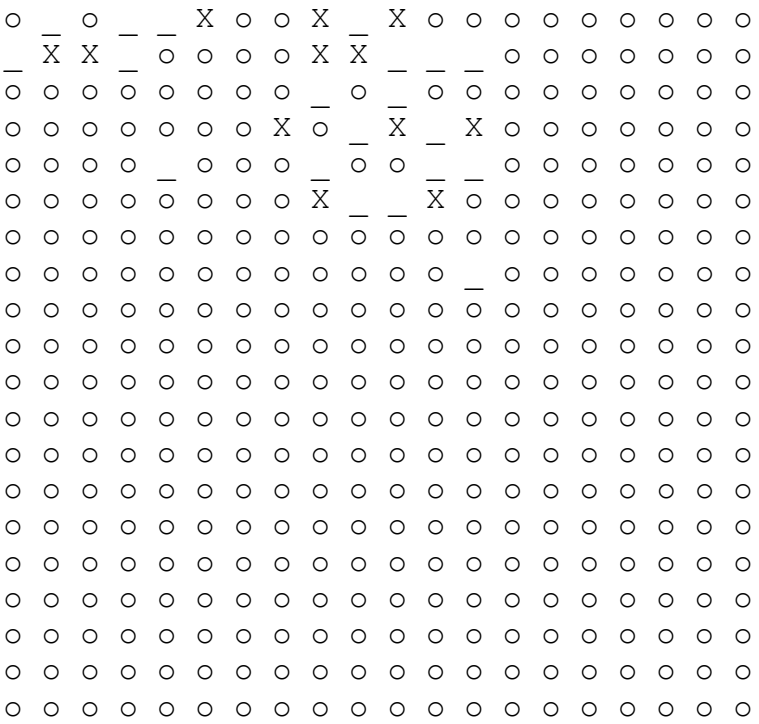


375 rabbits and 12 foxes

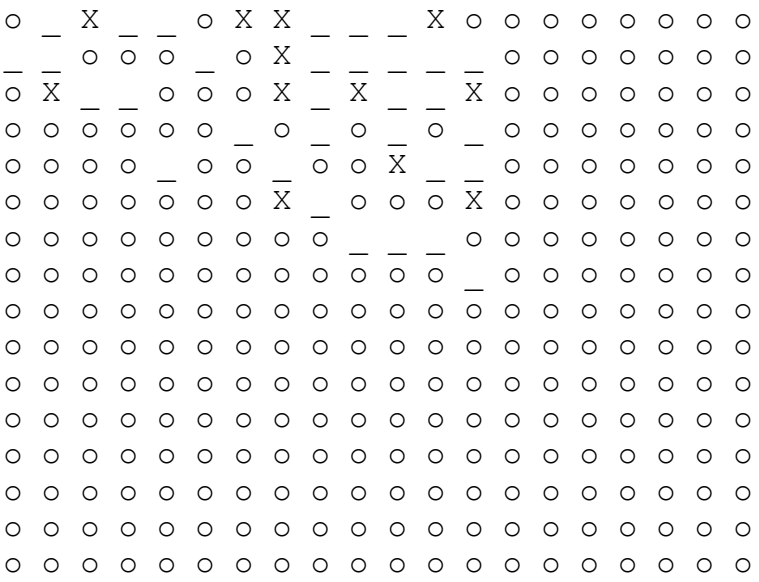


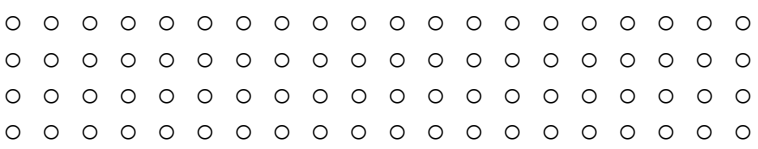


380 rabbits and 12 foxes

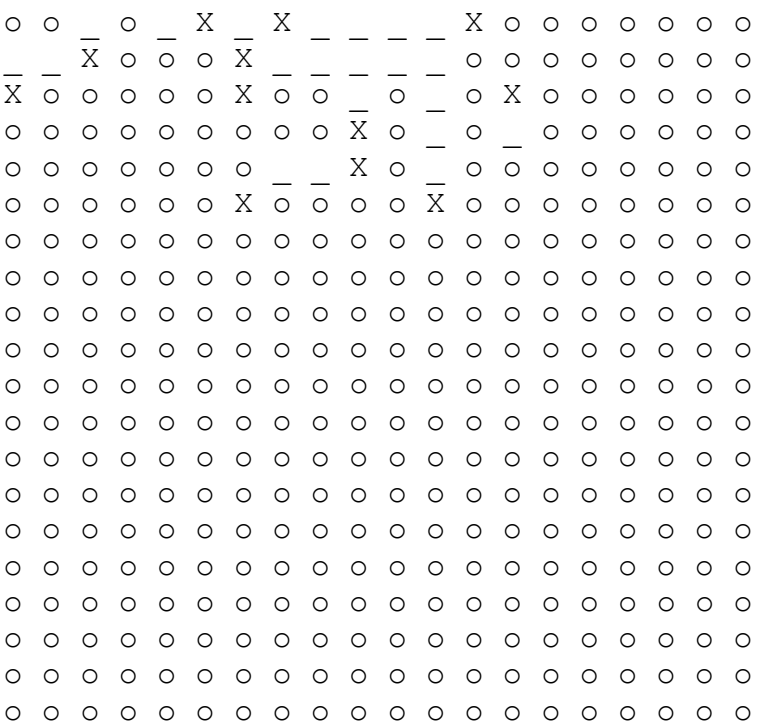


368 rabbits and 12 foxes

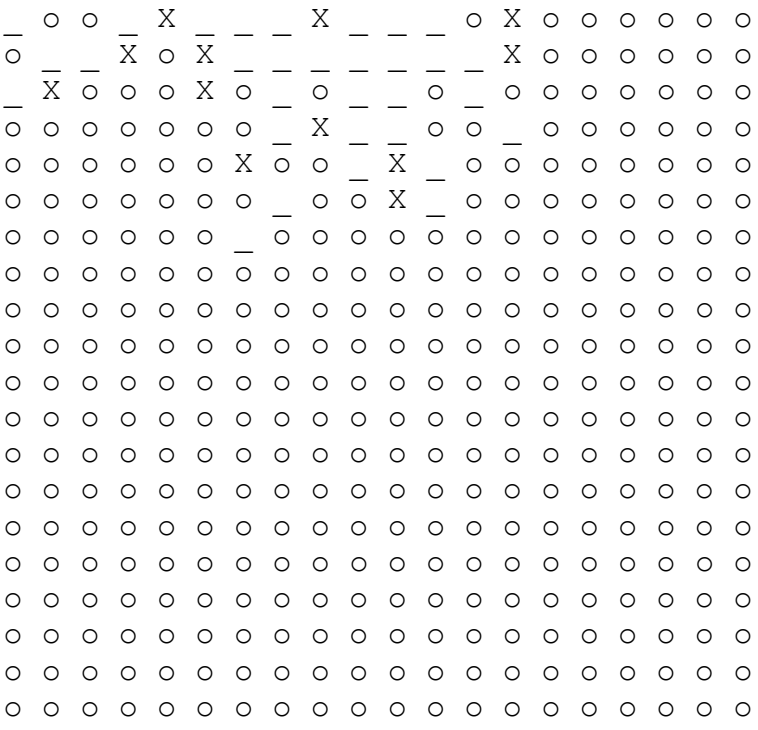




356 rabbits and 12 foxes



367 rabbits and 12 foxes



357 rabbits and 12 foxes