Προβλήματα εξάσκησης στην ύλη της Διαλέξης 6

1. Κατασκευάστε το αφηρημένο δεδομένο «mi», το οποίο λειτουργεί ως εξής:
   a. *Κατασκευαστής*: `make_a_mi(name)`
      Επιστρέφει ένα «mi» με όνομα που δίδεται στη συμβολοσειρά (`str`) `name`.
   b. *Συνάρτηση επιλογής*: `mi_do(m, command)`:
      Επιστρέφει το αποτέλεσμα της εντολής `command` όταν σταλθεί στο «mi» `m`.

   Οι εντολές που υποστηρίζουν τα «mi» είναι:
      i. `'who are you?'`: επιστρέφει το όνομα του.
      ii. `'what are you?'`: επιστρέφει τη συμβολοσειρά `'I am a mi!'`.

   Αν δεν δοθεί κάποια από τις παραπάνω εντολές επιστρέφει `'Command not recognized.'`.

   Παράδειγμα χρήσης από το διαδραστικό περιβάλλον του διερμηνευτή της Python:
```
>>> m = make_a_mi('Hugh')
>>> mi_do(m, 'what are you?')
'I am a mi!'
>>> mi_do(m, 'who are you?')
'Hugh'
>>> mi_do(m, 'sldkfdslf')
'Command not recognized.'
```

2. Κατασκευάστε το αφηρημένο δεδομένο `mi_army` το οποίο λειτουργεί ως εξής:
   a. *Κατασκευαστής*: `mi_army(mi_list)`
      Επιστρέφει αφηρημένο δεδομένο που αναπαριστά ένα στρατό από τα «mi» που βρίσκονται στη λίστα `mi_list`.
   b. *Συνάρτηση επιλογής*: `mi_army_do(ma, command)`:
      Επιστρέφει συμβολοσειρά που περιέχει τις απαντήσεις (χωριζόμενες με τελεία `'.'`) στην εντολή `command` όλων των «mi» του στρατού `ma`.

   Παράδειγμα χρήσης από το διαδραστικό περιβάλλον:
```
>>> m1 = make_a_mi('Hugh')
>>> m2 = make_a_mi('Lisa')
>>> m12 = mi_army([m1, m2])
>>> mi_army_do(m12, 'what are you?')
'I am a mi!.I am a mi!.'
>>> mi_army_do(m12, 'who are you?')
'Hugh.Lisa.'
>>> mi_army_do(m12, 'say something')
'Command not recognized.Command not recognized.'
```

3. Κατασκευάστε αφηρημένο δεδομένο `triplet` το οποίο αναπαριστά τριάδες τιμών, χρησιμοποιώντας λίστες 3 στοιχείων. Συγκεκριμένα, υλοποιήστε τις συναρτήσεις:
   a. *Κατασκευαστής*: `triplet(x, y, z)`
   b. *Συνάρτηση επιλογής*: `select(t, i)`
      Επιστρέφει το στοιχείο του triplet `t` το οποίο βρίσκεται στη θέση `i`, όπου `i`= `0`, `1` ή `2`.

Παράδειγμα χρήσης:

```
>>> t = triplet(10, 20, 30)
>>> print(select(t, 0), select(t, 1), select(t, 2))
10 20 30
```

4. Κατασκευάστε πάλι το αφηρημένο δεδομένο `triplet` χρησιμοποιώντας αυτή τη φορά συναρτήσεις 3 ορισμάτων. *Υπόδειξη: δείτε την υλοποίηση του ζεύγους pair με συναρτήσεις από τη Διάλεξη 6*.

5. Κατασκευάστε πάλι το αφηρημένο δεδομένο `triplet` χρησιμοποιώντας το αφηρημένο δεδομένο «pair» (συναρτήσεις `pair` και `select`) από τη Διάλεξη 6. *Υπόδειξη: πως μπορούμε να αναπαραστήσουμε μια τριπλέτα με ζεύγος (pair); ένα pair μπορεί να έχει στοιχείο ένα άλλο pair!*

6. Τα αφηρημένα δεδομένα τύπου `range` αναπαριστούν ακολουθίες αριθμών που απέχουν σταθερή απόσταση μεταξύ τους. Κατασκευάστε αφηρημένο δεδομένο `sequence` που αναπαριστά ακολουθίες αριθμών όπου οι διαδοχικοί όροι έχουν μεταβλητή απόσταση μεταξύ τους. Ο τύπος που καθορίζει την απόσταση διαδοχικών όρων δίδεται μέσω μιας συνάρτησης στο όρισμα του κατασκευαστή της `sequence`:
   a. *Κατασκευαστής*: `sequence(a, b, step_func)`
      Επιστρέφει αφηρημένο δεδομένο που αναπαριστά ακολουθία αριθμών με πρώτο όρο `a` και κάθε επόμενος όρος απέχει από τον προηγούμενο όρο, `x`, απόσταση `step_func(x)`. Ο τελευταίος όρος *δεν* είναι μεγαλύτερος ή ίσος του ορίσματος `b`.
   b. *Συνάρτηση επιλογής*: `select(s, i)`
      Επιστρέφει τον όρο της ακολουθίας `s` που βρίσκεται στη θέση `i`, όπου `i = 0, 1, 2, ...` (Ο $1^{ος}$ όρος βρίσκεται στη θέση `0`, ο $2^{ος}$ στη θέση `1` κοκ.)
   c. *Συνάρτηση επιλογής*: `length(s)`
      Επιστρέφει το πλήθος των όρων της ακολουθίας `s`

   Παράδειγμα χρήσης:

```
>>> r = sequence(0, 10, lambda x: 1) # παρόμοιο με range(10)
>>> length(r)
10
>>> select(r, 0)
0
>>> select(r, 9)
9
>>> even = sequence(0, 10, lambda x: 2) # 0, 2, 4, 6, 8
>>> length(even)
>>> select(even, 3)
6
>>> s = sequence(1, 17, lambda x: x)
>>> # Οι δυνάμεις του 2: 2^0,2^1,2^2,2^3,2^4
>>> select(s, 2)
8
```

7. Φτιάξτε το παιχνίδι της τρίλιζας χρησιμοποιώντας αφηρημένα δεδομένα όπως ταμπλό, θέση στο ταμπλό ή ότι άλλο τυχόν χρειαστεί.