ΠΛΗΡΟΦΟΡΙΚΗ Ι

ΕΡΓΑΣΤΗΡΙΟ 5

Θέμα εργαστηρίου: Διανύσματα

1. Να γραφεί συνάρτηση `reverse(n)` που να ζητάει από τον χρήστη την είσοδο *n* ακέραιων αριθμών και να τους εκτυπώνει στην οθόνη με την αντίθετη σειρά εισόδου.

2. Να γραφεί συνάρτηση `bubble` που να δέχεται διάνυσμα ακέραιων αριθμών, να τους ταξινομεί κατά αύξουσα σειρά με τον αλγόριθμο της φυσαλίδας (**bubble sort**) και να επιστρέφει το ταξινομημένο πλέον διάνυσμα.

Ο αλγόριθμος της φυσαλίδας είναι ο εξής:

1. Σύγκρινε κάθε ζευγάρι στοιχείων (1ο με 2ο, 2ο με 3ο, ..., (ν-1)ο με ν-οστό) και εάν τα στοιχεία κάποιου ζευγαριού είναι λάθος ταξινομημένα, ενάλλαξέ τα (swap)
2. Εάν έγινε έστω και μία εναλλαγή στοιχείων στο προηγούμενο βήμα, τότε επανέλαβε τη διαδικασία σύγκρισης των ζευγαριών (βήμα 1), διαφορετικά (δηλ., εάν δεν έγινε καμία εναλλαγή) τερμάτισε τον αλγόριθμο.
   (σε κάθε νέα επανάληψη του 1ου βήματος, το τελευταίο ζευγάρι της προηγούμενης επανάληψης της διαδικασίας δεν χρειάζεται να συγκριθεί, άρα συγκρίνεται πάντα ένα ζευγάρι λιγότερο)

3. Να γραφεί συνάρτηση `stats` που να δέχεται διάνυσμα πραγματικών αριθμών και να επιστρέφει τον μέγιστο, τον ελάχιστο, τον μέσο όρο όλων των αριθμών, τον μέσο όρων των θετικών και τον μέσο όρο των αρνητικών.
(να μη χρησιμοποιηθούν οι συναρτήσεις `min`, `max` και `sum` του MATLAB)

4. Δίνεται η ακόλουθη συνάρτηση `dice` που προσομοιώνει τη ρίψη ενός ζαριού:

```
function count = dice(rolls)
% Προσομοίωση ρίψης ζαριού
%   rolls: το πλήθος των ρίψεων του ζαριού
%   count: το διάνυσμα του πλήθους των αποτελεσμάτων της κάθε πλευράς,
%   δηλαδή, count(f) είναι το πόσες φορές εμφανίσθηκε η πλευρά f.

FACES= 6;              % ο αριθμός των πλευρών του ζαριού
count= zeros(1,FACES); % το διάνυσμα για την αποθήκευση του πλήθους κάθε πλευράς

for k= 1:rolls
    % ρίψη του ζαριού:
    face = ceil(rand(1)*FACES);
    % αύξηση του μετρητή της αντίστοιχης πλευράς:
    count(face) = count(face) + 1;
end
```

Αφού γίνει κατανοητή η λειτουργία της συνάρτησης αυτής,

α) να γραφεί συνάρτηση `rollDice(n,d)` που να προσομοιώνει τη ρίψη `d` ζαριών `n` φορές. Ορίζουμε το *αποτέλεσμα* της μίας ρίψης `d` ζαριών ως *το άθροισμα* των πλευρών που εμφανίσθηκαν σε κάθε ζάρι. Η συνάρτηση να επιστρέφει διάνυσμα `count` τέτοιο ώστε το `count(c)` να ισούται με τον αριθμό των φορών που έχει επιτευχθεί το αποτέλεσμα `c`. Να μη χρησιμοποιηθεί η συνάρτηση `sum` του MATLAB.
(Σκεφτείτε τι μεγέθους πρέπει να είναι το διάνυσμα `count`)

β) να γραφεί και να κληθεί η παρακάτω συνάρτηση `bargraph` (η οποία σχεδιάζει το ιστόγραμμα των αποτελεσμάτων) για διάφορες τιμές των `n` και `d`. Είσοδος της συνάρτησης είναι το διάνυσμα `count` που επιστρέφει η `rollDice`, καθώς και το πλήθος των ρίψεων των ζαριών (`n`), το οποίο θα πρέπει να ταυτίζεται με αυτό που χρησιμοποιήθηκε για την παραγωγή του συγκεκριμένου κάθε φορά `count`. Τι μορφή παίρνει το ιστόγραμμα για μεγάλες τιμές του `n`;

```
function bargraph(count,n)

d=length(count)/6;
bar(d:length(count), count(d:length(count)));
if (d==1)
   message = sprintf('%d ρίψεις ενός ζαριού', n);
else
   message = sprintf('%d ρίψεις %d ζαριών', n, d);
end
title(message);
xlabel('Αποτέλεσμα');  ylabel('Φορές');
```

γ) να γραφεί η αρχική συνάρτηση `dice` που δίνεται παραπάνω και να κληθεί η `bargraph` με αυτή αντί της `rollDice`. Ποια είναι τώρα η κατανομή των αποτελεσμάτων για μεγάλες τιμές του `n`;

5. Να γραφεί συνάρτηση `findElement(v,n,r)` που να επιστρέφει 1 εάν η τιμή `r` βρίσκεται στα πρώτα `n` στοιχεία του διανύσματος `v` και 0 σε διαφορετική περίπτωση.

6. Να γραφεί συνάρτηση `sequence(m)` που να παράγει μια αλληλουχία τυχαίων ακεραίων στο διάστημα [1, *m*] η οποία να σταματάει όταν κάποια τιμή επαναληφθεί για πρώτη φορά. Η συνάρτηση επιστρέφει τις τιμές που δημιουργήθηκαν (με τη σειρά που δημιουργήθηκαν), εκτός από την τελευταία επαναλαμβανόμενη τιμή.
(Να γίνει χρήση της συνάρτησης `findElement` της προηγούμενης άσκησης)