ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΜΕ ΡΥΤΗΟΝ (ΟΜΑΔΑ Α)

ΑΣΚΗΣΗ 1 [1 μονάδα]

α) Γράψτε έναν βρόχο while που κάνει ακριβώς το ίδιο με τον εξής βρόχο for:

```
for i in range(1,11):
        print "i =",  i
```

β) Ποιο θα είναι το αποτέλεσμα της εκτέλεσης των παρακάτω γραμμών κώδικα;

```
x = 3
while x >= 0:
    if x%2 == 0:
        print 'John'
    elif x - 3 < 0:
        print 'Mary'
    else:
        print 'Penny'
    x = x - 1
```

ΛΥΣΗ

α)

```
i = 1

while i < 11:

    print 'i =', i
    i += 1
```

β)

Penny
John
Mary
John

ΑΣΚΗΣΗ 2 [1 μονάδα]

α) Μετατρέψτε τον δεκαεξαδικό αριθμό $(1C2.B2)_{<16>}$ σε οκταδικό μέσω του δυαδικού συστήματος, δείχνοντας όλα τα βήματα της μετατροπής.

β) Εκτελέστε τις ακόλουθες πράξεις στα συστήματα που αναφέρονται για κάθε πράξη:

$(10.101)_{<2>} + (11.101)_{<2>}$

$(F1.12)_{<16>} + (6F.E2)_{<16>}$

ΛΥΣΗ

α) $(1C2.B2)_{<16>} = 0001\ 1100\ 0010\ .\ 1011\ 0010_{<2>} =$

$000\ 111\ 000\ 010\ .\ 101\ 100\ 100_{<2>} = (702.544)_{<8>}$

β)

Η εκτέλεση των προσθέσεων γίνεται ως ακολούθως:

μεταφορά:		111 1
		10.101
	+	11.101
		110.010

και τελικό αποτέλεσμα: $(110.010)_{<2>}$

μεταφορά: 11

F1.12

\+ 6F.E2

160.F4

ΑΣΚΗΣΗ 3 [1 μονάδα]

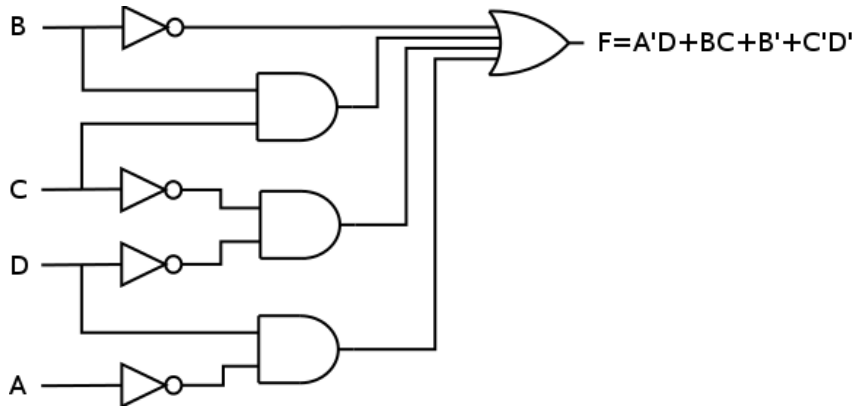
Έστω η συνάρτηση F = A' · D + B · C + B' + C' · D' (όπου οι τόνοι δηλώνουν συμπληρώματα).

i) Σχεδιάστε το κύκλωμα.

ii) Υπολογίστε για ποια είσοδο η F έχει έξοδο 0.

ΛΥΣΗ

i) Το κύκλωμα που προκύπτει από τη λογική συνάρτηση είναι το ακόλουθο:



ii)

A	**B**	**C**	**D**	**A'**	**B'**	**C'**	**D'**	**A' D**	**B C**	**C' D'**	**F**
0	0	0	0	1	1	1	1	0	0	1	1
0	0	0	1	1	1	1	0	1	0	0	1
0	0	1	0	1	1	0	1	0	0	0	1
0	0	1	1	1	1	0	0	1	0	0	1
0	1	0	0	1	0	1	1	0	0	1	1
0	1	0	1	1	0	1	0	1	0	0	1
0	1	1	0	1	0	0	1	0	1	0	1
0	1	1	1	1	0	0	0	1	1	0	1
1	0	0	0	0	1	1	1	0	0	1	1
1	0	0	1	0	1	1	0	0	0	0	1
1	0	1	0	0	1	0	1	0	0	0	1
1	0	1	1	0	1	0	0	0	0	0	1

A	B	C	D	A'	B'	C'	D'	A' D	B C	C' D'	F
1	1	0	0	0	0	1	1	0	0	1	1
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	0	0	1	0	1	0	1
1	1	1	1	0	0	0	0	0	1	0	1

ΑΣΚΗΣΗ 4 [1 μονάδα]

Βρείτε τι εκτελεί η παρακάτω αναδρομική συνάρτηση foobar με παράμετρο arg μια λίστα:

```
def foobar(arg):
    if arg == []:
        return arg
    else:
        return foobar(arg[1:]) + [arg[0]]
```

Χρησιμοποιήστε ως περιπτώσεις είσοδου τις εξής 3 λίστες:

arg1 = []
arg2 = [5,10,8]
arg3 = [1,2,2,1]

ΛΥΣΗ

Αν η arg είναι κενή λίστα, η foobar επιστρέφει την ίδια (κενή) λίστα. Αν η arg δεν είναι κενή λίστα, π.χ., arg = [5,10,8], τότε foobar([5,10,8]) = foobar([10,8]) + [5], αλλά foobar([10,8]) = foobar([8]) + [10] και foobar([8]) = foobar([]) + [8] = [] + [8] = [8]. Επομένως, foobar([5,10,8]) = [8] + [10] + [5] = [8,10,5], δηλαδή, η συνάρτηση foobar αντιστρέφει τα στοιχεία μιας λίστας.

ΑΣΚΗΣΗ 5 [1 μονάδα]

$$A = \begin{bmatrix} 4 & 2 & 1 \\ 8 & 11 & 1 \\ 2 & 1 & 4 \end{bmatrix},$$

Χρησιμοποιείστε την python ή το matlab για να βρείτε:

- Τον αντίστροφο του πίνακα A, τις ιδιοτιμές και τα ιδιοδιανύσματα του πίνακα A
- Την γραφική παράσταση της f(x)= 2*x+3*sin(x)+4*log(x+1)+1 στο διάστημα [0,2π]

ΛΥΣΗ

```
A=[4 2 1; 8 11 1; 2 1 4];
inv(A)
[p d]=eig(A)
x = 0:pi/100:2*pi;
y=2*x+3*sin(x)+4*log(x+1)+1;
plot(x,y)
```

ΑΣΚΗΣΗ 6 [1 μονάδα]

Γράψτε μια συνάρτηση recEq που παίρνει ως παράμετρο εισόδου ένα θετικό ακέραιο αριθμό n και επιστρέφει σε λίστα τους όρους της ακολουθίας:

$a_k = 5(-1)^k a_{k-1}$, $a_1 = 3$, για k = 2, …, n.

ΛΥΣΗ

```
def recEq(n):
    a = []
    k = 1
    while k <= n:
        if k == 1:
            a.append(3)
        else:
```

```
        a.append((-1)**k * 5 * a[-1])
    k += 1
  return a
```

ΑΣΚΗΣΗ 7 [1 μονάδα]

Γράψτε μια συνάρτηση ConvertEurosToDollars που παίρνει τρεις παραμέτρους εισόδου: amount (που εκφράζει το ποσό σε ευρώ), rate (που εκφράζει την ισοτιμία ευρώ-δολάριου), charge (που εκφράζει ένα ποσοστό χρέωσης της συναλλαγής). Η συνάρτηση ConvertEurosToDollars πρέπει να επιστρέφει το καθαρό ποσό σε δολάρια ενός ποσού ευρώ μετά την αφαίρεση της χρέωσης της συναλλαγής. Η χρέωση της συναλλαγής είναι amount*charge αν το ποσό σε ευρώ είναι μεγαλύτερο των 10 ευρώ.

ΛΥΣΗ

```
def ConvertEurosToDollars(amount, rate, charge):
  total = amount * rate
  if amount < 10:
    fee = 0
  else:
    fee = charge / 100. * amount
  return total - fee
```

ΑΣΚΗΣΗ 8 [1 μονάδα]

Γράψτε μια συνάρτηση hotelStay με τέσσερις παραμέτρους εισόδου: price (που εκφράζει την τιμή του δωματίου του ξενοδοχείου ανά διανυχτέρευση), days (που εκφράζει τις διανυχτερεύσεις στο ξενοδοχείο), tax (που εκφράζει τον ΦΠΑ επί τοις εκατό στη συνολική τιμή) και fee (που εκφράζει μια επιπρόσθετη χρέωση ανά μέρα παραμονής αφού συμπεριληφθεί στο ποσό συνολικής πληρωμής και ο ΦΠΑ). Η συνάρτηση hotelStay πρέπει να επιστρέφει το τελικό ποσό που στοιχίζει η παραμονή στο ξενοδοχείο. Διευκρινίζεται ότι η τιμή του fee δεν προστίθεται στο τελικό ποσό αν οι μέρες διαμονής είναι περισσότερες των 6, ενώ λαμβάνεται υποψή υποδιπλασιασμένη (δηλ. fee/2) αν οι μέρες διαμονής είναι από 4 έως και 6.

ΛΥΣΗ

```
def hotelStay(price, days, tax, fee):
        if days > 6:
                fee = 0
        elif days > 3:
                fee = fee/2.
        total = price * days
        totalPlusTax = total * (1 + tax/100.)
        return totalPlusTax + fee * days
```

ΑΣΚΗΣΗ 9 [2 μονάδες]

Έστω ότι τα αποτελέσματα κάποιων αγώνων ποδοσφαίρου δίνονται με μια λίστα της μορφής

```
agwnes = ['aa-bb: 3-1', 'aa-cc: 1-1', 'bb-cc: 0-2', 'cc-dd: 2-2', 'dd-aa: 2-1'],
```

όπου aa, bb, cc, dd είναι ομάδες (παρατηρείστε ότι αμέσως μετά το ':' υπάρχει κενό).

Γράψτε μια συνάρτηση bathmologia με παράμετρο εισόδου τη λίστα agwnes, η οποία να επιστρέφει ένα λεξικό με κλειδιά τις ομάδες της λίστας agwnes και τιμές την βαθμολογία τους. Δίνεται ότι για κάθε νίκη μια ομάδα (όταν δηλ. έχει πετύχει περισσότερα γκολ από την αντίπαλη ομάδα) κερδίζει 3 βαθμούς για κάθε ισοπαλία 1 βαθμό (όταν δηλ. έχει πετύχει ίσο πλήθος γκολ σε σχέση με την αντίπαλη ομάδα) ενώ δεν κερδίζει κανένα βαθμό σε περίπτωση ήττας.

ΛΥΣΗ

```
def bathmologia(agwnes):
  scores = {}
  for i in agwnes:
    tl=i.split(": ")
    tl1=tl[0].split("-")
    if tl1[0] not in scores:
      scores[tl1[0]]=0
    if tl1[1] not in scores:
```

```
            scores[tl1[1]]=0
        tl2=tl[1].split("-")
        if tl2[0] > tl2[1]:
            scores[tl1[0]] += 3
        elif tl2[0] < tl2[1]:
            scores[tl1[1]] += 3
        else:
            scores[tl1[0]] += 1
            scores[tl1[1]] += 1
    return scores
```