Λύσεις
Πανελλαδικών
Θεμάτων
Γ,Δ
2000-2017

ανάπτυξη

Εφαρμογών σε

προγραμματιστικό

περιβάλλον με

πάϊθον

Αναστάσιος
Χατζηπαπαδόπουλος

αεππп

ανάπτυξη εφαρμογών σε προγραμματιστικό περιβάλλον με πάϊθον

Λύσεις Πανελλαδικών Θεμάτων Γ & Δ έτη 2000-2017

ISBN 978-960-93-9926-5

Αναστάσιος Χατζηπαπαδόπουλος
Εκπαιδευτικός Πληροφορικής

ISBN 978-960-93-9926-5

Αναστάσιος Χατζηπαπαδόπουλος
Εκπαιδευτικός Πληροφορικής
chatzipapwork@gmail.com
http://users.sch.gr/chatzipap

Περιεχόμενα

Πρόλογος

Αγαπητοί συνάδελφοι και μαθητές,

2000–2017…

18 ολόκληρα έτη ΑΕΠΠ, τα άρρενα νεογέννητα της εποχής εκείνης σήμερα θα παρουσιάζονταν ως νεοσύλλεκτοι ή θα ζητούσαν αναβολή στράτευσης λόγω σπουδών.

Κατά πολλούς, στη διάρκεια αυτών των ετών το μάθημα έκλεισε έναν κύκλο ενηλικίωσης, κατ' άλλους έκλεισε περισσότερους από έναν.

Το περιεχόμενο του μαθήματος, οι ασκήσεις, η θεματολογία των πανελλαδικών εξετάσεων, έζησαν για 18 έτη κάτω από το στενό και ασαφές πλαίσιο της ψευδοΓΛΩΣΣΑΣ του ΑΕΠΠ. Το περιεχόμενο αναζητά «αναβάθμιση» με ενδεχόμενη χρήση μίας ζωντανής γλώσσας η οποία ακόμα και κάτω από τους περιορισμούς και τις δεσμεύσεις που λογικά θα έπρεπε να καθοριστούν με σαφήνεια σε ένα νέο διδακτικό πακέτο (ΑΠΣ/Βιβλίο) θα έφερναν ένα αέρα ανανέωσης, δημιουργώντας ένα σύγχρονο μάθημα Αλγοριθμικής/Αρχών Προγραμματισμού.

Γεγονός όμως είναι ότι, όλα αυτά τα έτη δημιουργήθηκε μία αξιοπρεπής βάση δεδομένων 80 περίπου πανελλαδικών θεμάτων διαφόρων βαθμών δυσκολίας.

Γεννήθηκε έτσι η ιδέα της επίλυσής τους με ένα σύγχρονο εργαλείο προγραμματισμού, αυτό της γλώσσας python η οποία, όχι μόνο χρησιμοποιείται σε πολύ μεγάλο βαθμό σε παγκόσμια κλίμακα για την εισαγωγή στην αλγοριθμική, αλλά αποτελεί και ένα σύγχρονο εργαλείο ανάπτυξης και προγραμματισμού. Επίσης έχει ενταχθεί σε ΑΠΣ και σε διδακτικά εγχειρίδια και χρησιμοποιείται για τη διδασκαλία των Αρχών Προγραμματισμού στα **ΕΠΑ.Λ. από το 2015**.

Στο βιβλίο αυτό περιέχονται προτεινόμενες/ενδεικτικές λύσεις στα πανελλαδικά θέματα Γ & Δ (έτη 2000 – 2017) του μαθήματος της Ανάπτυξης Εφαρμογών σε Προγραμματιστικό Περιβάλλον με τη χρήση όμως όχι της ψευδοΓΛΩΣΣΑΣ αλλά της Python (έκδοση 2.7.10) με τον τρόπο που αυτή παρουσιάζεται στα διδακτικά εγχειρίδια της Β' και Γ' Τάξης του Τομέα Πληροφορικής των ΕΠΑΛ.

Έγινε προσπάθεια να περιοριστεί η χρήση βιβλιοθηκών, εντολών, μεθόδων κλπ. που δεν παρουσιάζονται στο αντίστοιχο βιβλίο των ΕΠΑΛ.

Οι λύσεις έχουν αποδοθεί, με ένα μικρό βαθμό ελευθερίας ως προς την αρχική εκφώνηση των θεμάτων προκειμένου να επιτευχθεί συμβατότητα με τη γλώσσα python.

Τα προβλήματα και οι λύσεις τους δεν αποσκοπούν και δεν προορίζονται στο σύνολό τους χωρίς παρεμβάσεις (πχ αφαίρεση υποερωτημάτων) για μαθητές των ΕΠΑΛ. Αυτό διότι αρκετά από αυτά αφενός δεν συμβαδίζουν με την διδακτέα ύλη των ΕΠΑΛ, αφετέρου ο βαθμός δυσκολίας τους δεν κρίνεται κατάλληλος για αυτή τη βαθμίδα. Μπορούν όμως με κατάλληλες τροποποιήσεις ή αλλαγές στην εκφώνηση να αποτελέσουν αντικείμενο δραστηριοτήτων για μαθητές/εκπαιδευτικούς διαφόρων βαθμίδων ή κατευθύνσεων.

Το περιεχόμενο του βιβλίου (εκφωνήσεις/ενδ. Λύσεις) θα το βρείτε και σε online-interactive μορφή στο Διαδικτυακό τόπο http://users.sch.gr/chatzipap

Ελπίζω το παρόν βιβλίο να αποτελέσει ένα χρήσιμο βοήθημα για μαθητές και εκπαιδευτικούς.

Παρατηρήσεις / προτάσεις στο chatzipapwork@gmail.com (θέμα: aeppp).

Για τη συγγραφή και εκτύπωση των σεναρίων χρησιμοποιήθηκε το λογισμικό Notepad ++ έκδοση 7.2.2 (https://notepad-plus-plus.org/).

Ενώ για την εκτέλεσή τους το περιβάλλον των:

- IDLE (Python έκδοση 2.7.10) https://www.python.org/ &
- Pycharm Community Edition 2016.3 https://www.jetbrains.com/pycharm/

☺ Happy Learning

Αναστάσιος Χατζηπαπαδόπουλος

Εκπαιδευτικός Πληροφορικής

ΛΥΣΕΙΣ ΘΕΜΑΤΩΝ

Γ & Δ

ΑΕΠΠ

ΜΕ ΓΛΩΣΣΑ PYTHON

ΈΤΗ 2000 – 2017

ΘΕΜΑ Γ_0_2000 (Ολυμπιακοί Αγώνες Σίδνεϋ)

Σε τρεις διαφορετικούς αγώνες πρόκρισης για την Ολυμπιάδα του Σίδνεϋ στο άλμα εις μήκος ένας αθλητής πέτυχε τις επιδόσεις a,b,c. Να αναπτύξετε αλγόριθμο ο οποίος:

α) να διαβάζει τις τιμές των επιδόσεων a,b,c

β) να υπολογίζει και να εμφανίζει τη μέση τιμή των παραπάνω τιμών

γ) να εμφανίζει το μήνυμα «ΠΡΟΚΡΙΘΗΚΕ», αν η παραπάνω μέση τιμή είναι μεγαλύτερη των 8 μέτρων.

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:        2000_0_C
# Περιγραφή:    Ολυμπιακοί Αγώνες Σίδνεϋ
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ------------------------------------------------------------------

# α)
a, b, c = input("Δώσε τις επιδόσεις a,b,c = ")

# β)
mo = (a + b + c) * 1.0 / 3
print mo

# γ)
if mo > 8:
    print "Προκρίθηκες"
else:
    print "Απέτυχες"
```

ΘΕΜΑ Δ_0_2000 (Λογαριασμός Κινητής Τηλεφωνίας)

Μια εταιρεία κινητής τηλεφωνίας ακολουθεί ανά μήνα την πολιτική τιμών που φαίνεται στον παρακάτω πίνακα:

Πάγιο 1500 δραχμές

Χρόνος τηλεφωνημάτων (δευτερόλεπτα)	Χρονοχρέωση (δραχμές/δευτερόλεπτο)
1-500	1,5
501-800	0,9
801 και άνω	0,5

Να αναπτύξετε αλγόριθμο ο οποίος:

α) να διαβάζει τη χρονική διάρκεια των τηλεφωνημάτων ενός συνδρομητή σε διάστημα ενός μήνα

β) να υπολογίζει τη μηνιαία χρέωση του συνδρομητή

γ) να εμφανίζει (τυπώνει) τη λέξη «ΧΡΕΩΣΗ» και τη μηνιαία χρέωση του συνδρομητή.

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:       2000_0_D
# Περιγραφή:   Λογαριασμός Κινητής Τηλεφωνίας
# Ενδλύση:     TasosChatzipapadopoulos
# Python vs:   2.7.10
# Copyright:   (c) 2018
# Url:         http://users.sch.gr/chatzipap
# ------------------------------------------------------------------

# α)
time = input("Δώσε το χρόνο μηνιαίας ομιλίας = ")

# β)
# Μη κλιμακωτή χρέωση
if time <= 500:
    charge = 1.5
elif time < 800:
    charge = 0.9
else:
    charge = 0.5

# γ)
total = 1500 + charge * time
print "charge=", total
```

ΘΕΜΑ Γ_1_2000 (Κατανάλωση Ρεύματος)

Μια οικογένεια κατανάλωσε Χ Kwh (κιλοβατώρες) ημερήσιου ρεύματος και Υ Kwh νυχτερινού ρεύματος. Το κόστος ημερήσιου ρεύματος είναι 30 δρχ. ανά Kwh και του νυχτερινού 15 δρχ. ανά Kwh Να αναπτύξετε έναν αλγόριθμο ο οποίος:

α. να διαβάζει τα Χ, Υ

β. να υπολογίζει και να εμφανίζει το συνολικό κόστος της κατανάλωσης ρεύματος της οικογένειας

γ. να εμφανίζει το μήνυμα ΥΠΕΡΒΟΛΙΚΗ ΚΑΤΑΝΑΛΩΣΗ, αν το συνολικό κόστος είναι μεγαλύτερο από 100.000 δραχμές.

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:       2000_1_C
# Περιγραφή:   Κατανάλωση Ρεύματος
# Ενδλύση:     TasosChatzipapadopoulos
# Python vs:   2.7.10
# Copyright:   (c) 2018
# Url:         http://users.sch.gr/chatzipap
# ------------------------------------------------------------------

# α)
x, y = input('Δώσε κατανάλωση ημερησίου, νυχτερινού τιμολογίου = ')

# β)
cons = x * 30 + y * 15
print "Κατανάλωση=", cons

# γ)
if cons > 100000:
    print "Υπερβολική ΚΑτανάλωση"
```

ΘΕΜΑ Δ_1_2000 (Προσαρμογή Τελικού Βαθμού)

Ο τελικός βαθμός ενός μαθητή σένα μάθημα υπολογίζεται με βάση την προφορική και τη γραπτή βαθμολογία του με την ακόλουθη διαδικασία: Αν η διαφορά των δυο βαθμών είναι μεγαλύτερη από πέντε (5) μονάδες, τότε ο προφορικός βαθμός προσαρμόζεται (δηλαδή αυξάνεται ή μειώνεται) έτσι, ώστε η αντίστοιχη διαφορά να μειωθεί στις τρεις (3) μονάδες, αλλιώς ο προφορικός βαθμός παραμένει αμετάβλητος. Ο τελικός βαθμός είναι ο μέσος όρος των δυο βαθμών.

Παράδειγμα προσαρμογής προφορικού βαθμού: Αν ο γραπτός βαθμός είναι 18 και ο προφορικός 11, τότε ο προφορικός γίνεται 15, ενώ, αν ο γραπτός είναι 10 και ο προφορικός 19, τότε ο προφορικός γίνεται 13. Να αναπτύξετε έναν αλγόριθμο ο οποίος:

α. να διαβάζει τους δυο βαθμούς

β. να υπολογίζει τον τελικό βαθμό σύμφωνα με την παραπάνω διαδικασία

γ. να εμφανίζει τον τελικό βαθμό και/ αν αυτός είναι μεγαλύτερος ή ίσος του 10/ το μήνυμα ΠΡΟΑΓΕΤΑΙ/ αλλιώς το μήνυμα ΑΠΟΡΡΙΠΤΕΤΑΙ.

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:        2000_1_D
# Περιγραφή:    Προσαρμογή Τελικού Βαθμού
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ------------------------------------------------------------------

# α)
write, oral = input("Δώσε βαθμολογία γραπτά, προφορικά = ")
d = oral - write
if write > oral:
    d = write - oral

# β)
if d > 5:
    if write > oral:
        oral = write - 3
    else:
        oral = write + 3
mo = (write + oral) * 1.0 / 2
print write, oral, mo

# γ)
if mo >= 10:
    print "Πέτυχες"
else:
    print "Απέτυχες"
```

ΘΕΜΑ Γ_0_2001 (Ελάχιστο στοιχείο δισδιάστατης λίστας)

Δίνεται πίνακας Π δύο διαστάσεων, που τα στοιχεία του είναι ακέραιοι αριθμοί με Ν γραμμές και Μ στήλες. Να αναπτύξετε αλγόριθμο που να υπολογίζει το ελάχιστο στοιχείο του πίνακα.

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:       2001_0_C
# Περιγραφή:   Ελάχιστο στοιχείο δισδιάστατης λίστας
# Ενδλύση:     TasosChatzipapadopoulos
# Python vs:   2.7.10
# Copyright:   (c) 2018
# Url:         http://users.sch.gr/chatzipap
# ------------------------------------------------------------------

# όπου πίνακας θεωρούμε μία λίστα
# έστω διάστασης 4x3

# η λίστα θα γεμίζει με τυχαίους ακεραίους από 1..100
import random
a_list = []
for i in range(4):
    row = []
    for j in range(3):
        x = random.randint(1, 100)
        row.append(x)
    a_list.append(row)

print a_list

# υπολογισμός ελάχιστου στοιχείου
xmin = a_list[0][0]

for i in range(4):
    for j in range(3):
        if a_list[i][j] < xmin:
            xmin = a_list[i][j]

print xmin
```

ΘΕΜΑ Δ_0_2001 (Συλλογή Γυαλιού, Χαρτιού, Αλουμινίου)

Σε ένα πρόγραμμα περιβαλλοντικής εκπαίδευσης συμμετέχουν 20 σχολεία. Στα πλαίσια αυτού του προγράμματος, εθελοντές μαθητές των σχολείων, που συμμετέχουν στο πρόγραμμα, μαζεύουν ποσότητες τριών υλικών (γυαλί, χαρτί και αλουμίνιο).

Να αναπτύξετε έναν αλγόριθμο, ο οποίος:

α. να διαβάζει τις ποσότητες σε κιλά των παραπάνω υλικών που μάζεψαν οι μαθητές σε κάθε σχολείο

β. να υπολογίζει τη συνολική ποσότητα σε κιλά του κάθε υλικού που μάζεψαν οι μαθητές σε όλα τα σχολεία

γ. αν η συνολική ποσότητα του χαρτιού που μαζεύτηκε από όλα τα σχολεία είναι λιγότερη των 1000 κιλών, να εμφανίζεται το μήνυμα «Συγχαρητήρια». Αν η ποσότητα είναι από 1000 κιλά και πάνω, αλλά λιγότερο από 2000, να εμφανίζεται το μήνυμα «Δίνεται έπαινος» και τέλος αν η ποσότητα είναι από 2000 κιλά και πάνω να εμφανίζεται το μήνυμα «Δίνεται βραβείο».

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:       2001_0_D
# Περιγραφή:   Συλλογή Γυαλιού, Χαρτιού, Αλουμινίου
# ΕνδΛύση:     TasosChatzipapadopoulos
# Python vs:   2.7.10
# Copyright:   (c) 2018
# Url:         http://users.sch.gr/chatzipap
# ------------------------------------------------------------------

# α), β)
s_glass = s_paper = s_alu = 0
for i in range(5):
    glass, paper, alu = input("Δώσε ποσότητα γυαλιού, χαρτιού, αλουμινίου = ")
    s_glass += glass
    s_paper += paper
    s_alu += alu

print s_glass, s_paper, s_alu

# γ)
if s_paper < 1000:
    print "Συγχαρητήρια"
elif s_paper < 2000:
    print "Δίνεται έπαινος"
else:
    print "Δίνεται βραβείο"
```

ΘΕΜΑ Γ_1_2001 (Τυποποιητής Πορτοκαλιών)

Να αναπτύξετε αλγόριθμο ο οποίος υλοποιεί τη λειτουργία ενός αυτόματου τυποποιητή πορτοκαλιών που είναι η παρακάτω:

Για κάθε πορτοκάλι που εισάγεται στον τυποποιητή, διαβάζεται η τιμή του βάρους του (Β) και η διάμετρος του (Δ). Το πορτοκάλι κατατάσσεται ανάλογα με το βάρος και τη διάμετρο του ως εξής:

Αν 100 < Β < 150 και 8 < Δ < 10, τότε τυπώνεται το μήνυμα "πρώτη διαλογή".

Αν 6 < Δ < 8, τότε, ανεξαρτήτως βάρους, τυπώνεται το μήνυμα "δεύτερη διαλογή".

Σε κάθε άλλη περίπτωση τυπώνεται το μήνυμα "χυμοποίηση".

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:       2001_1_C
# Περιγραφή:   Τυποποιητής Πορτοκαλιών
# ΕνδΛύση:     TasosChatzipapadopoulos
# Python vs:   2.7.10
# Copyright:   (c) 2018
# Url:         http://users.sch.gr/chatzipap
# ------------------------------------------------------------------

w, d = input("Δώσε βάρος, διάμετρο πορτοκαλιού = ")
if 100 < w < 150 and 8 < d < 10:
    print "πρώτη διαλογή"
elif 6 < d < 8:
    print "δεύτερη διαλογή"
else:
    print "χυμοποίηση"
```

ΘΕΜΑ Δ_1_2001 (Διεθνείς αγώνες στίβου Ρίψεις ακοντίου)

Κατά τη διάρκεια Διεθνών Αγώνων Στίβου στον ακοντισμό έλαβαν μέρος δέκα (10) αθλητές. Κάθε αθλητής έκανε έξι (6) έγκυρες ρίψεις που καταχωρούνται ως επιδόσεις σε μέτρα. Να αναπτύξετε αλγόριθμο, ο οποίος:

α. εισάγει σε πίνακα δύο διαστάσεων τις επιδόσεις όλων των αθλητών

β. υπολογίζει και καταχωρεί σε μονοδιάστατο πίνακα την καλύτερη από τις επιδόσεις κάθε αθλητή

γ. ταξινομεί τις καλύτερες επιδόσεις των αθλητών που καταχωρήθηκαν στο μονοδιάστατο πίνακα

δ. βρίσκει την καλύτερη επίδοση του αθλητή που πήρε το χάλκινο μετάλλιο (τρίτη θέση).

Παρατήρηση: Υποθέτουμε ότι όλες οι επιδόσεις είναι μεταξύ τους διαφορετικές.

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:        2001_1_D
# Περιγραφή:    Διεθνείς αγώνες στίβου Ρίψεις ακοντίου
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ------------------------------------------------------------------


# α)
javelin = []
for i in range(10):
    athlete = []
    for i in range(6):
        throw = input("Δώσε τη ρίψη ακοντίου σε μέτρα = ")
        athlete.append(throw)
    javelin.append(athlete)

#  β)
best = []
for i in range(10):
    xmax = javelin[i][0]
    for j in range(6):
        if javelin[i][j] > xmax:
            xmax = javelin[i][j]
    best.append(xmax)

print 'Καλύτερη Επίδοση Αθλητή'
print best

# γ)
n = len(best)
for i in range(n-1):
    for j in range(n-1, i, -1):
        if best[j] > best[j-1]:
            best[j], best[j-1] = best[j-1], best[j]

print 'Καλύτερες Επιδόσεις'
```

```
print best

# δ)
print 'Χάλκινο Μετάλλιο'
print best[2]
```

ΘΕΜΑ Γ_0_2002 (Σύστημα Αντιτίμου πληρωμής διοδίων)

Με το νέο σύστημα πληρωμής των διοδίων, οι οδηγοί των τροχοφόρων έχουν τη δυνατότητα να πληρώνουν το αντίτιμο των διοδίων με ειδική μαγνητική κάρτα. Υποθέστε ότι υπάρχει μηχάνημα το οποίο διαθέτει είσοδο για την κάρτα και φωτοκύτταρο. Το μηχάνημα διαβάζει από την κάρτα το υπόλοιπο των χρημάτων και το αποθηκεύει σε μία μεταβλητή Υ και, με το φωτοκύτταρο, αναγνωρίζει τον τύπο του τροχοφόρου και το αποθηκεύει σε μία μεταβλητή Τ. Υπάρχουν τρεις τύποι τροχοφόρων: δίκυκλα (Δ), επιβατικά (Ε) και φορτηγά (Φ), με αντίτιμο διοδίων 1, 2 και 3 ευρώ αντίστοιχα.

Να αναπτύξετε αλγόριθμο, ο οποίος:

α. ελέγχει τον τύπο του τροχοφόρου και εκχωρεί στη μεταβλητή Α το αντίτιμο των διοδίων, ανάλογα με τον τύπο του τροχοφόρου

β. ελέγχει την πληρωμή των διοδίων με τον παρακάτω τρόπο. Αν το υπόλοιπο της κάρτας επαρκεί για την πληρωμή του αντιτίμου των διοδίων, αφαιρεί το ποσό αυτό από την κάρτα. Αν η κάρτα δεν έχει υπόλοιπο, το μηχάνημα ειδοποιεί με μήνυμα για το ποσό που πρέπει να πληρωθεί. Αν το υπόλοιπο δεν επαρκεί, μηδενίζεται η κάρτα και δίνεται με μήνυμα το ποσό που απομένει να πληρωθεί.

```
# ----------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ----------------------------------------------------------------
# Όνομα:       2002_0_C
# Περιγραφή:   Σύστημα Αντιτίμου πληρωμής διοδίων
# ΕνδΛύση:     TasosChatzipapadopoulos
# Python vs:   2.7.10
# Copyright:   (c) 2018
# Url:         http://users.sch.gr/chatzipap
# ----------------------------------------------------------------

account = input("Δώσε υπόλοιπο χρημάτων στον λογαριασμό = ")
vehicle = raw_input("Δώσε τον τύπο οχήματος moto(m), car(c), truck(t) =")

# α)
if vehicle == "m":
    pay = 1
elif vehicle == "c":
    pay = 2
elif vehicle == "t":
    pay = 3


# β)
if account >= pay:
    print "ΟΚ Περάστε Ελεύθερα"
    account -= pay
    print "Υπόλοιπο Λογαριασμού = ", account
elif account == 0:
    print "Αποτυχία"
    print "Πρέπει να πληρώσετε ολόκληρο το αντίτιμο", pay
else:
    print "Πρέπει να πληρώσετε ", pay - account, "ακόμα"
    account = 0
    print "Υπόλοιπο Λογαριασμού ", account
```

ΘΕΜΑ Δ_0_2002 (Διατήρηση προϊόντων σε αποθήκες)

Μια εταιρεία αποθηκεύει είκοσι (20) προϊόντα σε δέκα (10) αποθήκες. Να γράψετε πρόγραμμα στη γλώσσα προγραμματισμού "ΓΛΩΣΣΑ", το οποίο:

α. περιέχει τμήμα δήλωσης των μεταβλητών του προγράμματος

β. εισάγει σε μονοδιάστατο πίνακα τα ονόματα των είκοσι προϊόντων

γ. εισάγει σε πίνακα δύο διαστάσεων Π[20,10] την πληροφορία που αφορά στην παρουσία ενός προϊόντος σε μια αποθήκη (καταχωρούμε την τιμή 1 στην περίπτωση που υπάρχει το προϊόν στην αποθήκη και την τιμή 0, αν το προϊόν δεν υπάρχει στην αποθήκη).

δ. υπολογίζει σε πόσες αποθήκες βρίσκεται το κάθε προϊόν

ε. τυπώνει το όνομα κάθε προϊόντος και το πλήθος των αποθηκών στις οποίες υπάρχει το προϊόν.

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:        2002_0_D
# Περιγραφή:    Διατήρηση προϊόντων σε αποθήκες
# ΕνδΛύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ------------------------------------------------------------------

# β)
product = []
for i in range(20):
    product.append(raw_input("Δώσε την ονομασία του προϊόντος = "))

# γ)
storages = []
for i in range(20):
    storage = []
    print "Εισαγωγή Προϊόντος ", product[i]
    for j in range(10):
        print "Αποθήκη με αροθμό ", j
        storage.append(input("Δώσε (1)για ύπαρξη & (0)για απουσία "))
    storages.append(storage)

# δ)
for i in range(20):
    count = 0
    for j in range(10):
        if storages[i][j] == 1:
            count += 1
    print product[i], " Υπάρχει σε ", count, " Αποθήκες"
```

ΘΕΜΑ Γ_1_2002 (Δημιουργία Διμοιριών σε κέντρο Νεοσύλλεκτων)

Σε ένα κέντρο νεοσύλλεκτων υπάρχει η πρόθεση να δημιουργηθούν δύο ειδικές διμοιρίες. Η διμοιρία Α θα αποτελείται από νεοσύλλεκτους πτυχιούχους τριτοβάθμιας εκπαίδευσης, ηλικίας από 24 έως και 28 χρόνων. Η διμοιρία Β θα αποτελείται από νεοσύλλεκτους απόφοιτους δευτεροβάθμιας εκπαίδευσης, ηλικίας από 18 έως και 24 χρόνων. Οι υπόλοιποι νεοσύλλεκτοι δεν κατατάσσονται σε καμία από αυτές τις διμοιρίες. Να αναπτύξετε αλγόριθμο ο οποίος:

α. διαβάζει το ονοματεπώνυμο, την ηλικία και έναν αριθμό που καθορίζει το επίπεδο σπουδών του νεοσύλλεκτου και παίρνει τιμές από 1 έως 3 (1: τριτοβάθμια εκπαίδευση, 2: δευτεροβάθμια εκπαίδευση, 3: κάθε άλλη περίπτωση)

β. εκτυπώνει:

i) το ονοματεπώνυμο του νεοσύλλεκτου

ii) το όνομα της διμοιρίας (Α ή Β), εφόσον ο νεοσύλλεκτος κατατάσσεται σε μία από αυτές.

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:        2002_1_C
# Περιγραφή:    Δημιουργία Διμοιριών σε κέντρο Νεοσύλλεκτων
# ΕνδΛύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ------------------------------------------------------------------

# α)
name = raw_input("Δώσε το όνομα του νεοσύλλεκτου = ")
age = input("Δώσε την ηλικία του νεοσύλλεκτου = ")
edu = input("Δώσε το επίπεδο σπουδών του 1,2,3 = ")

# β)
if edu == 1:
    if age >= 24 and age <= 28:
        print name, "platoon A"
elif edu ==2:
    if age >= 18 and age <= 24:
        print name, "platoon B"
```

ΘΕΜΑ Δ_1_2002 (Εισπράξεις σε αλυσίδα Ξενοδοχείων)

Μια αλυσίδα ξενοδοχείων έχει 5 ξενοδοχεία. Σε ένα μονοδιάστατο πίνακα ΞΕΝΟΔΟΧΕΙΑ[5] καταχωρούνται τα ονόματα των ξενοδοχείων. Σε ένα άλλο δισδιάστατο πίνακα ΕΙΣΠΡΑΞΕΙΣ[5,12] καταχωρούνται οι εισπράξεις κάθε ξενοδοχείου για κάθε μήνα του έτους 2001, έτσι ώστε στην Ι γραμμή καταχωρούνται οι εισπράξεις του i ξενοδοχείου.

Να αναπτύξετε αλγόριθμο, ο οποίος:

α. διαβάζει τα στοιχεία των δύο πινάκων

β. εκτυπώνει το όνομα κάθε ξενοδοχείου και τις ετήσιες εισπράξεις του για το έτος 2001

γ. εκτυπώνει το όνομα του ξενοδοχείου με τις μεγαλύτερες εισπράξεις για το έτος 2001.

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:        2002_1_D
# Περιγραφή:    Εισπράξεις σε αλυσίδα Ξενοδοχείων
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ------------------------------------------------------------------

# α)
hotels = []
money = []
for i in range(5):
    hotels.append(raw_input("Δώσε το όνομα του ξενοδοχείου = "))
    money.append([])
    for j in range(12):
        money[i].append(input("Δώσε τις μηνιαίες εισπράξεις του = "))

# β)
total = []
for i in range(5):
    xsum = 0
    for j in range(12):
        xsum += money[i][j]
    total.append(xsum)

# γ)
xmax = 0
name = ''
for i in range(5):
    if total[i] > xmax:
        xmax = total[i]
        name = hotels[i]

print "Μεγαλύτερες Εισπράξεις"
print name, xmax
```

ΘΕΜΑ Γ_0_2003 (Δείκτης BMI(ΔΜΣ) ύψους/μάζας σώματος)

Ο Δείκτης Μάζας του ανθρώπινου Σώματος (ΔΜΣ) υπολογίζεται από το βάρος (Β) σε χλγ.

και το ύψος (Υ) σε μέτρα με τον τύπο ΔΜΣ=Β/Υ^2. Ο ανωτέρω τύπος ισχύει για άτομα άνω των 18 ετών. Το άτομο ανάλογα με την τιμή του ΔΜΣ χαρακτηρίζεται σύμφωνα με τον παρακάτω πίνακα:

ΔΜΣ<18,5 «αδύνατο άτομο»

18,5<ΔΜΣ<25 «κανονικό άτομο»

25<ΔΜΣ<30 «βαρύ άτομο»

30<ΔΜΣ «υπέρβαρο άτομο»

Να γράψετε αλγόριθμο ο οποίος:

α. να διαβάζει την ηλικία, το βάρος και το ύψος του ατόμου

β. εάν η ηλικία είναι μεγαλύτερη των 18 ετών, τότε να υπολογίζει το ΔΜΣ να ελέγχει την τιμή του ΔΜΣ από τον ανωτέρω πίνακα και να εμφανίζει τον αντίστοιχο χαρακτηρισμό

γ. εάν η ηλικία είναι μικρότερη ή ίση των 18 ετών, τότε να εμφανίζει το μήνυμα «δεν ισχύει ο δείκτης ΔΜΣ».

Παρατήρηση: Θεωρήστε ότι το βάρος, το ύψος και η ηλικία είναι θετικοί αριθμοί.

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:        2003_0_C
# Περιγραφή:    Δείκτης BMI(ΔΜΣ) ύψους/μάζας σώματος
# ΕνδΛύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ------------------------------------------------------------------

# α)
age = input("Δώσε την ηλικία σου = ")

# α), β), γ)
if age > 18:
    weight = input("Δώσε το βάρος σου = ")
    height = input("Δώσε το ύψος σου = ")
    bmi = weight / (height * 0.01) ** 2
    print bmi,
    if bmi < 18.5:
        print "αδύνατο άτομο"
    elif bmi < 25:
        print "κανονικό άτομο"
    elif bmi < 30:
        print "βαρύ άτομο"
    else:
        print "υπέρβαρο άτομο"
else:
    print "Δεν ισχύει ο ΔΜΣ για την ηλικία σου"
```

ΘΕΜΑ Δ_0_2003 (Μηνιαίες Εισπράξεις Αλυσίδας Κινηματογράφων)

Μια αλυσίδα κινηματογράφων έχει δέκα αίθουσες. Τα ονόματα των αιθουσών καταχωρούνται σε ένα μονοδιάστατο πίνακα και οι μηνιαίες εισπράξεις κάθε αίθουσας για ένα έτος καταχωρούνται σε πίνακα δύο διαστάσεων.

Να γράψετε αλγόριθμο ο οποίος:

α. να διαβάζει τα ονόματα των αιθουσών

β. να διαβάζει τις μηνιαίες εισπράξεις των αιθουσών αυτού του έτους

γ. να υπολογίζει τη μέση μηνιαία τιμή των εισπράξεων για κάθε αίθουσα

δ. να βρίσκει και να εμφανίζει τη μικρότερη μέση μηνιαία τιμή

ε. να βρίσκει και να εμφανίζει το όνομα ή τα ονόματα των αιθουσών που έχουν την ανωτέρω μικρότερη μέση μηνιαία τιμή.

Παρατήρηση : Θεωρήστε ότι οι μηνιαίες εισπράξεις είναι θετικοί αριθμοί.

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:        2003_0_D
# Περιγραφή:    Μηνιαίες Εισπράξεις Αλυσίδας Κινηματογράφων
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ------------------------------------------------------------------

# α), β)
# Εισαγωγή στοιχείων στις λίστες
names = []
money = []
for i in range(10):
    names.append(raw_input("Δώσε το όνομα του κινηματογράφου = "))
    money.append([])
    for j in range(12):
        money[i].append(input("Δώσε τις μηνιαίες Εισπράξεις = "))

# γ)
# Υπολογισμός μέσου όρου εισπράξεων και αποθήκευση σε λίστα
mo = []
for i in range(10):
    xsum = 0.0
    for j in range(12):
        xsum += money[i][j]
    mo.append(xsum / 10)
    print names[i], mo[i]

# δ)
# Εύρεση μικρότερων εισπράξεων
xmin = mo[0]
for i in range(10):
    if mo[i] < xmin:
        xmin = mo[i]
```

```
print "The smallest monthly taking is = ", xmin

# ε)
# Εύρεση αιθουσών με τις μικρότερες εισπράξεις
print "Οι κινηματογράφοι με τις μικρότερες μηνιαίες εισπράξεις είναι:"
for i in range(10):
    if xmin == mo[i]:
        print names[i], mo[i]
```

ΘΕΜΑ Γ_1_2003 (Τιμολόγιο Κατανάλωσης νερού)

Κάποια δημοτική αρχή ακολουθεί την εξής τιμολογιακή πολιτική για την κατανάλωση νερού ανά μήνα. Χρεώνει πάγιο ποσό 2 ευρώ και εφαρμόζει κλιμακωτή χρέωση σύμφωνα με τον παρακάτω πίνακα :

Κατανάλωση σε κυβικά μέτρα	Χρέωση ανά κυβικό
από 0 έως και 5	δωρεάν
από 5 έως και 10	0,5 ευρώ
από 10 έως και 20	0,7 ευρώ
από 20 και άνω	1,0 ευρώ

Στο ποσό που προκύπτει από την αξία του νερού και το πάγιο υπολογίζεται ο Φ.Π.Α. με συντελεστή 18%. Το τελικό ποσό προκύπτει από την άθροιση της αξίας του νερού, το πάγιο, το Φ.Π.Α. και το δημοτικό φόρο που είναι 5 ευρώ. Να γράψετε αλγόριθμο ο οποίος:

α. Να διαβάζει τη μηνιαία κατανάλωση του νερού.

β. Να υπολογίζει την αξία του νερού που καταναλώθηκε σύμφωνα με την παραπάνω τιμολογιακή πολιτική.

γ. Να υπολογίζει το Φ.Π.Α.

δ. Να υπολογίζει και να εκτυπώνει το τελικό ποσό.

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:        2003_1_C
# Περιγραφή:    Τιμολόγιο Κατανάλωσης νερού
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ------------------------------------------------------------------

# α)
cons = input("Δώσε την κατανάλωση σε m3 = ")

# β)
if cons <= 5:
    m3 = 0
elif cons <= 10:
    m3 = 0 + (cons - 5) * 0.5
elif cons <= 20:
    m3 = 0 + 5 * 0.5 + (cons - 10) * 0.7
else:
    m3 = 0 + 5 * 0.5 + 10 * 0.7 + (cons - 20) * 1.0

# γ), δ)
fpa = (m3 + 2) * 0.18
total = m3 + fpa + 5
print "Θα πρέπει να πληρώσετε = ", total
```

ΘΕΜΑ Δ_1_2003 (Στατιστικά παικτών σε αγώνες μπάσκετ)

Κατά τη διάρκεια του πρωταθλήματος μπάσκετ μια ομάδα που αποτελείται από δώδεκα παίκτες έδωσε είκοσι αγώνες, στους οποίους συμμετείχαν όλοι οι παίκτες.

Να αναπτύξετε στο τετράδιό σας αλγόριθμο ο οποίος :

α. Να διαβάζει τα ονόματα των παικτών και να τα αποθηκεύει σε μονοδιάστατο πίνακα.

β. Να διαβάζει τους πόντους που σημείωσε κάθε παίκτης σε κάθε αγώνα και να τους αποθηκεύει σε πίνακα δύο διαστάσεων.

γ. Να υπολογίζει για κάθε παίκτη το συνολικό αριθμό πόντων του σε όλους τους αγώνες και το μέσο όρο πόντων ανά αγώνα.

δ. Να εκτυπώνει τα ονόματα των παικτών της ομάδας και το μέσο όρο πόντων του κάθε παίκτη ταξινομημένα με βάση το μέσο όρο τους κατά φθίνουσα σειρά.

Παρατήρηση: Σε περίπτωση ισοβαθμίας δε μας ενδιαφέρει η σχετική σειρά των παικτών.

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:       2003_1_D
# Περιγραφή:   Στατιστικά παικτών σε αγώνες μπάσκετ
# Ενδλύση:     TasosChatzipapadopoulos
# Python vs:   2.7.10
# Copyright:   (c) 2018
# Url:         http://users.sch.gr/chatzipap
# ------------------------------------------------------------------

# α), β)
# Εισαγωγή στοιχείων στις λίστες
names = []
points = []
for i in range(12):
    names.append(raw_input("Δώσε το όνομα του παίκτη = "))
    points.append([])
    for j in range(20):
        points[i].append(input("Δώσε τους πόντους που πέτυχε = "))

# γ)
# Εύρεση μέσο όρου πόνταων κάθε παίκτη
mo = []
for i in range(12):
    xsum = 0.0
    for j in range(20):
        xsum += points[i][j]
    mo.append(xsum / 20)

# δ)
# Ταξινόμηση ονομάτων βάση του ΜΟ των πόντων
for i in range(11):
    for j in range(11, i, -1):
        if mo[j] > mo[j-1]:
            mo[j], mo[j-1] = mo[j-1], mo[j]
            names[j], names[j-1] = names[j-1], names[j]
```

```
print 'Οι παρακάτω παίκτες πέτυχαν ΜΟ πόντων'
for i in range(12):
    print names[i], mo[i]
```

ΘΕΜΑ Γ_0_2004 (Κοστολόγηση επιστολών ταχυδρομείου)

Μία εταιρεία ταχυδρομικών υπηρεσιών εφαρμόζει για τα έξοδα αποστολής ταχυδρομικών επιστολών εσωτερικού και εξωτερικού, χρέωση σύμφωνα με τον παρακάτω πίνακα:

Βάρος επιστολής σε γραμμάρια	Χρέωση εσωτερικού σε Ευρώ	Χρέωση εξωτερικού σε Ευρώ
από 0 έως και 500	2,0	4,8
από 500 έως και 1000	3,5	7,2
από 1000 έως και 2000	4,6	11,5

Για παράδειγμα τα έξοδα αποστολής μιας επιστολής βάρους 800 γραμμαρίων και προορισμού εσωτερικού είναι 3,5 Ευρώ.

Να γράψετε αλγόριθμο ο οποίος :

α. Να διαβάζει το βάρος της επιστολής.

β. Να διαβάζει τον προορισμό της επιστολής. Η τιμή ΕΣ δηλώνει προορισμό εσωτερικού και η τιμή ΕΞ δηλώνει προορισμό εξωτερικού.

γ. Να υπολογίζει τα έξοδα αποστολής ανάλογα με τον προορισμό και το βάρος της επιστολής.

δ. Να εκτυπώνει τα έξοδα αποστολής.

Παρατήρηση. Θεωρήστε ότι ο αλγόριθμος δέχεται τιμές για το βάρος μεταξύ του 0 και του 2000 και για τον προορισμό μόνο τις τιμές "ΕΣ" και "ΕΞ".

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:        2004_0_C
# Περιγραφή:    Κοστολόγηση επιστολών ταχυδρομείου
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ------------------------------------------------------------------

# α)
weight = input("Δώσε το βάρος του φακέλλου σε γρ. = ")

# β)
country = raw_input("Προορισμός (in)Εσωτερικό, (out)Εξωτερικό = ")

# γ)
if country == 'in':
    if weight <= 500:
        price = 2.0
    elif weight <= 1000:
        price = 3.5
    else:
        price = 4.6
elif country == 'out':
    if weight <= 500:
        price = 4.8
    elif weight <= 1000:
```

```
        price = 7.2
    else:
        price = 11.5

# δ)
print "Κόστος Αποστολής = ", price
```

ΘΕΜΑ Δ_0_2004 (Βαθμολογίες σε Ολυμπιάδα Πληροφορικής)

Για την πρώτη φάση της Ολυμπιάδας Πληροφορικής δήλωσαν συμμετοχή 500 μαθητές. Οι μαθητές διαγωνίζονται σε τρεις γραπτές εξετάσεις και βαθμολογούνται με ακέραιους βαθμούς στη βαθμολογική κλίμακα από 0 έως και 100.

Να γράψετε αλγόριθμο ο οποίος:

α. Να διαβάζει τα ονόματα των μαθητών και να τα αποθηκεύει σε μονοδιάστατο πίνακα.

β. Να διαβάζει τους τρεις βαθμούς που έλαβε κάθε μαθητής και να τους αποθηκεύει σε δισδιάστατο πίνακα.

γ. Να υπολογίζει το μέσο όρο των βαθμών του κάθε μαθητή.

δ. Να εκτυπώνει τα ονόματα των μαθητών και δίπλα τους το μέσο όρο των βαθμών τους ταξινομημένα με βάση τον μέσο όρο κατά φθίνουσα σειρά.

Σε περίπτωση ισοβαθμίας η σειρά ταξινόμησης των ονομάτων να είναι αλφαβητική.

ε. Να υπολογίζει και να εκτυπώνει το πλήθος των μαθητών με το μεγαλύτερο μέσο όρο.

Παρατήρηση: Θεωρείστε ότι οι βαθμοί των μαθητών είναι μεταξύ του 0 και του 100 και ότι τα ονόματα των μαθητών είναι γραμμένα με μικρά γράμματα.

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:        2004_0_D
# Περιγραφή:    Βαθμολογίες σε Ολυμπιάδα Πληροφορικής
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ------------------------------------------------------------------

# α)
# Εισαγωγή ονομάτων μαθητών
names = []
for i in range(500):
    names.append(raw_input("Δώσε το όνομα του μαθητή = "))

# β)
# Εισαγωγή βαθμολογίας μαθητών
grades = []
for i in range(500):
    a, b, c = input("Δώσε τις 3 βαθμολογίες του a,b,c = ")
    grades.append([a, b, c])

# γ)
# Υπολογισμός μέσων όρων βαθμολογίας
mos = []
for i in range(500):
    mo = (grades[i][0] + grades[i][1] + grades[i][2]) / 3.0
    mos.append(mo)

# δ)
# Εμφάνιση ταξινομημένης λίστας ονομάτων/βαθμολογίας
for i in range(499):
```

```
    for j in range(499, i, -1):
        if mos[j] > mos[j-1]:
            mos[j], mos[j-1] = mos[j-1], mos[j]
            names[j], names[j-1] = names[j-1], names[j]
        elif mos[j] == mos[j-1]:
            if names[j] < names[j-1]:
                names[j], names[j-1] = names[j-1], names[j]

print "Ονόματα μαθητών / βαθμολογίες"
for i in range(500):
    print names[i], mos[i]

# ε)
# Εμφάνιση πλήθους μαθητών με βαθμολογία πάνω από τον μέσο όρο
max_mo = mos[0]
count = 0
for i in range(500):
    if mos[i] >= max_mo:
        count += 1
print "Πλήθος μαθητών πάνω από το ΜΟ = ", count
```

ΘΕΜΑ Γ_1_2004 (Βαθμολόγηση & Αναβαθμολόγηση γραπτού εξετάσεων)

Σε κάποια εξεταστική δοκιμασία κάθε γραπτό αξιολογείται αρχικά από δύο βαθμολογητές και υπάρχει περίπτωση το γραπτό να χρειάζεται αναβαθμολόγηση από τρίτο βαθμολογητή. Στην περίπτωση αναβαθμολόγησης ο τελικός βαθμός υπολογίζεται ως εξής:

i. Αν ο βαθμός του τρίτου βαθμολογητή είναι ίσος με το μέσο όρο (Μ.Ο.) των βαθμών των δύο πρώτων βαθμολογητών, τότε ο τελικός βαθμός είναι ο Μ.Ο.

ii. Αν ο βαθμός του τρίτου βαθμολογητή είναι μικρότερος από το μικρότερο βαθμό (MIN) των δύο πρώτων βαθμολογητών, τότε ο τελικός βαθμός είναι ο MIN.

iii. Διαφορετικά, ο τελικός βαθμός είναι ο μέσος όρος του βαθμού του τρίτου βαθμολογητή με τον πλησιέστερο προς αυτόν βαθμό των δύο πρώτων βαθμολογητών.

Να αναπτύξετε αλγόριθμο υπολογισμού του τελικού βαθμού ενός γραπτού με αναβαθμολόγηση, ο οποίος:

α. να διαβάζει τους βαθμούς του πρώτου, του δεύτερου και του τρίτου βαθμολογητή ενός γραπτού.

β. να υπολογίζει και να εκτυπώνει το μεγαλύτερο (MAX) και το μικρότερο (MIN) από τους βαθμούς του πρώτου και του δεύτερου βαθμολογητή.

γ. να υπολογίζει και να εκτυπώνει τον τελικό βαθμό του γραπτού σύμφωνα με την παραπάνω διαδικασία.

Παρατήρηση: Θεωρήστε ότι και οι τρεις βαθμοί είναι θετικοί ακέραιοι αριθμοί και δεν απαιτείται έλεγχος των δεδομένων.

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:       2004_1_C
# Περιγραφή:   Βαθμολόγηση & Αναβαθμολόγηση γραπτού εξετάσεων
# Ενδλύση:     TasosChatzipapadopoulos
# Python vs:   2.7.10
# Copyright:   (c) 2018
# Url:         http://users.sch.gr/chatzipap
# ------------------------------------------------------------------

# α)
a, b, c = input("Δώσε 3 βαθμολογίες a,b,c = ")
xmax = a
xmin = b

# β)
if xmax < xmin:
    xmax = b
    xmin = a
print "Μεγαλύτερος = ", xmax, "Μικρότερος = ", xmin

# γ)
mo = (a + b) / 2.0
if mo == c:
    final = mo
elif c < xmin:
    final = xmin
# Επίλυση χωρίς χρήση απόλυτης τιμής
else:
```

```
    d1 = c - xmin  # Σίγουρα c > xmin
    if c > xmax:
        d2 = c - xmax
    else:
        d2 = xmax - c
    if d1 < d2:
        final = (xmin + c) / 2.0
    else:
        final = (xmax + c) / 2.0

print "Τελικός βαθμός = ", final
```

ΘΕΜΑ Δ_1_2004 (Εκλογές Ευρωπαϊκού Κοινοβουλίου)

Σε κάποια χώρα της Ευρωπαϊκής Ένωσης διεξάγονται εκλογές για την ανάδειξη των μελών του Ευρωπαϊκού Κοινοβουλίου. Θεωρήστε ότι μετέχουν 15 συνδυασμοί κομμάτων, οι οποίοι θα μοιραστούν 24 έδρες σύμφωνα με το ποσοστό των έγκυρων ψηφοδελτίων που έλαβαν. Κόμματα που δεν συγκεντρώνουν ποσοστό έγκυρων ψηφοδελτίων τουλάχιστον ίσο με το 3% του συνόλου των έγκυρων ψηφοδελτίων δεν δικαιούνται έδρα. Για κάθε κόμμα, εκτός του πρώτου κόμματος, ο αριθμός των εδρών που θα λάβει υπολογίζεται ως εξής: Το ποσοστό των έγκυρων ψηφοδελτίων πολλαπλασιάζεται επί 24 και στη συνέχεια το γινόμενο διαιρείται με το άθροισμα των ποσοστών όλων των κομμάτων που δικαιούνται έδρα. Το ακέραιο μέρος του αριθμού που προκύπτει είναι ο αριθμός των εδρών που θα λάβει το κόμμα. Το πρώτο κόμμα λαμβάνει τις υπόλοιπες έδρες.

Να γράψετε αλγόριθμο ο οποίος:

α. να διαβάζει και να αποθηκεύει σε μονοδιάστατους πίνακες τα ονόματα των κομμάτων και τα αντίστοιχα ποσοστά των έγκυρων ψηφοδελτίων τους.

β. να εκτυπώνει τα ονόματα και το αντίστοιχο ποσοστό έγκυρων ψηφοδελτίων των κομμάτων που δεν έλαβαν έδρα.

γ. να εκτυπώνει το όνομα του κόμματος με το μεγαλύτερο ποσοστό έγκυρων ψηφοδελτίων.

δ. να υπολογίζει και να εκτυπώνει το άθροισμα των ποσοστών όλων των κομμάτων που δικαιούνται έδρα.

ε. να εκτυπώνει τα ονόματα των κομμάτων που έλαβαν έδρα και τον αντίστοιχο αριθμό των εδρών τους.

Παρατηρήσεις: α) Υποθέτουμε ότι δεν υπάρχουν δύο κόμματα που να έχουν το ίδιο ποσοστό έγκυρων ψηφοδελτίων.

β) Μπορείτε να χρησιμοποιήσετε τη συνάρτηση A_M(x) που επιστρέφει το ακέραιο μέρος του πραγματικού αριθμού x.

γ) Τα ποσοστά να θεωρηθούν επί τοις εκατό (%).

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:        2004_1_D
# Περιγραφή:    Εκλογές Ευρωπαϊκού Κοινοβουλίου
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ------------------------------------------------------------------

# α)
party = []
votes = []
# Εισαγωγή στοιχείων
for i in range(15):
    party.append(raw_input("Δώσε το όνομα του κόμματος = "))
    votes.append(input("Δώσε το ποσοστό % των ψήφων = "))

# β)
# Κόμματα χωρίς έδρα
print "Κόμματα χωρίς έδρα"
for i in range(15):
```

```
    if votes[i] < 3:
        print party[i], votes[i]

# γ)
# Πρώτο κόμμα
imax = 0
xmax = 0
for i in range(15):
    if votes[i] > xmax:
        xmax = votes[i]
        imax = i

print "Πρώτο κόμμα ", party[imax], votes[imax]

# δ)
# Άθροισμα ποσοστών
xsum = 0
for i in range(15):
    if votes[i] >= 3:
        xsum += votes[i]

print "% Κομμάτων χωρίς έδρα ", xsum

# ε)
# Υπολογισμός εδρών των κομμάτων
total_seats = 0
for i in range(15):
    if i != imax:
        if votes[i] >= 3:
            seats = int(votes[i] * 24 / xsum)
            total_seats += seats
            print party[i], seats

print "Το 1ο κόμμα έλαβε ", 24 - total_seats, " έδρες"
```

ΘΕΜΑ Γ_0_2005 (Εύρεση τρέχων μέσου μίας λίστας)

Δίνεται πίνακας Α[Ν] ακέραιων και θετικών αριθμών, καθώς και πίνακας Β[Ν-1] πραγματικών και θετικών αριθμών.

Να γραφεί αλγόριθμος, ο οποίος να ελέγχει αν κάθε στοιχείο Β[i] είναι ο μέσος όρος των στοιχείων Α[i] και Α[i+1], δηλαδή αν Β[i] = (Α[i] + Α[i+1])/2. Σε περίπτωση που ισχύει, τότε να εμφανίζεται το μήνυμα «Ο πίνακας Β είναι ο τρέχων μέσος του Α», διαφορετικά να εμφανίζεται το μήνυμα «Ο πίνακας Β δεν είναι ο τρέχων μέσος του Α».

Για παράδειγμα:

Έστω ότι τα στοιχεία του πίνακα Α είναι: 1, 3, 5, 10, 15 και ότι τα στοιχεία του πίνακα Β είναι: 2, 4, 7.5, 12.5. Τότε ο αλγόριθμος θα εμφανίσει το μήνυμα «Ο πίνακας Β είναι ο τρέχων μέσος του Α», διότι 2 = (1+3)/2, 4=(3+5)/2, 7.5= (5+10)/2, 12.5=(10+15)/2.

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:        2005_0_C
# Περιγραφή:    Εύρεση τρέχων μέσου μίας λίστας
# ΕνδΛύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ------------------------------------------------------------------

# Ορισμός λιστών
a = [1,3,5,10,15]
b = [2,4,7.5,12.5]

# Έλεγχος τρέχων μέσου
n = len(a)
mesos = True
for i in range(n-1):    # Για μεγάλες λίστες μπορούμε να χρησιμοποιήσουμε while
    if (a[i] + a[i+1]) / 2.0 != b[i]:
        mesos = False

print 'Τρέχων μέσος = ', mesos
```

ΘΕΜΑ Δ_0_2005 (Εξετάσεις με ερωτήσεις πολλαπλής επιλογής)

Σ' ένα διαγωνισμό συμμετέχουν 100 υποψήφιοι. Κάθε υποψήφιος διαγωνίζεται σε 50 ερωτήσεις πολλαπλής επιλογής.

Να αναπτύξετε αλγόριθμο που να κάνει τα παρακάτω:

α. Να καταχωρεί σε πίνακα ΑΠ[100,50] τα αποτελέσματα των απαντήσεων του κάθε υποψηφίου σε κάθε ερώτηση. Κάθε καταχώρηση μπορεί να είναι μόνο μία από τις παρακάτω:

i. Σ αν είναι σωστή η απάντηση

ii. Λ αν είναι λανθασμένη η απάντηση και

iii. Ξ αν ο υποψήφιος δεν απάντησε.

Να γίνεται έλεγχος των δεδομένων εισόδου.

β. Να βρίσκει και να τυπώνει τους αριθμούς των ερωτήσεων που παρουσιάζουν το μεγαλύτερο βαθμό δυσκολίας, δηλαδή έχουν το μικρότερο πλήθος σωστών απαντήσεων.

γ. Αν κάθε Σ βαθμολογείται με 2 μονάδες, κάθε Λ με -1 μονάδα και κάθε Ξ με 0 μονάδες τότε

i. Να δημιουργεί ένα μονοδιάστατο πίνακα ΒΑΘ[100], κάθε στοιχείο του οποίου θα περιέχει αντίστοιχα τη συνολική βαθμολογία ενός υποψηφίου.

ii. Να τυπώνει το πλήθος των υποψηφίων που συγκέντρωσαν βαθμολογία μεγαλύτερη από 50.

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:        2005_0_D
# Περιγραφή:    Εξετάσεις με ερωτήσεις πολλαπλής επιλογής
# ΕνδΛύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ------------------------------------------------------------------

# α)
# Εισαγωγή στοιχείων στις λίστες και έλεγχος δεδομένων
answers = []
for i in range(100):
    answers.append([])
    for j in range(50):
        a = raw_input("Δώστε την απάντησή σας (T)rue, (F)alse, (B)lank= ")
        while a.lower() not in ['t', 'f', 'b']:
            print "Λάθος Δεδομένα ξαναπροσπαθήστε..."
            a = raw_input("Δώστε την απάντησή σας (T)rue, (F)alse, (B)lank= ")
        answers[i].append(a.lower())

# β)
# Εύρεση ορθών απαντήσεων
correct = []
for i in range(100):
    correct.append(0)
    for j in range(50):
        if answers[i][j] == 't':
            correct[i] += 1
```

```
# Εύρεση και εμφάνιση δυσκολότερων ερωτήσεων
xmin = correct[0]
for i in range(50):
    if correct[i] < xmin:
        xmin = correct[i]

print "Οι δυσκολότερες ερωτήσεις ήταν ..."
for i in range(50):
    if correct[i] == xmin:
        print i,

# γ)
# Ευρεση και εμφάνιση βαθμολογίας
grades = []
for i in range(100):
    grades.append(0)
    for j in range(50):
        if answers[i][j] == 't':
            grades[i] += 2
        elif answers[i][j] == 'f':
            grades[i] += -1

# Εύρεση και εμφάνιση υποψηφίων με βαθ μεγαλύτερη από 50
count = 0
for i in range(100):
    if grades[i] >= 50:
        count += 1

print "Πάνω από 50 ήταν =", count
```

ΘΕΜΑ Γ_1_2005 (Βαθμολογίες υποψηφίων ΑΣΕΠ)

Εκατό (100) υποψήφιοι του ΑΣΕΠ διαγωνίζονται σε τρία μαθήματα για την κάλυψη θέσεων του Δημοσίου. Να γραφεί κύριο πρόγραμμα σε ΓΛΩΣΣΑ που να κάνει τα παρακάτω:

α) Διαβάζει τα ονόματα των 100 υποψηφίων του ΑΣΕΠ και τη βαθμολογία καθενός υποψηφίου σε τρία διαφορετικά μαθήματα. (Θεωρήστε ότι η βαθμολογία κάθε μαθήματος είναι από 1 έως 20).

β) Βρίσκει και τυπώνει τον ελάχιστο και τον μέγιστο βαθμό καθενός υποψηφίου στα τρία μαθήματα που εξετάστηκε.

γ) Να γραφεί υποπρόγραμμα, το οποίο να καλείται από το κύριο πρόγραμμα, για τον υπολογισμό και την εκτύπωση του μέσου όρου κάθε υποψηφίου στα τρία μαθήματα που διαγωνίστηκε.

```
# ----------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ----------------------------------------------------------------
# Όνομα:        2005_1_C
# Περιγραφή:    Βαθμολογίες υποψηφίων ΑΣΕΠ
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ----------------------------------------------------------------

# γ)
# Συνάρτηση υπολογισμού μικρότερου και μεγαλύτερου από 3 αριθμούς


def min_max(a, b, c):
    xmin = a
    if b < xmin:
        xmin = b
    if c < xmin:
        xmin = c
    xmax = a
    if b > xmax:
        xmax = b
    if c > xmax:
        xmax = c
    return xmin, xmax

# γ)
# Συνάρτηση υπολογισμού μέσου όρου βαθμολογίας


def grades_average(a, b, c):
    return (a + b + c) / 3.0


# α)
# Εισαγωγή στοιχείων κλήσεις και εκτυπώσεις
grades = []
for i in range(100):
    grades.append([])
    for j in range(3):
```

```
        grades[i].append(input("Δώστε τη βαθμολογία = "))
    print "Ελάχιστος, Μέγιστος = ", min_max(grades[i][0], grades[i][1],
grades[i][2]),
    print "Μέσος Όρος Βαθμολογίας = ", grades_average(grades[i][0], grades[i][1],
grades[i][2])
```

ΘΕΜΑ Δ_1_2005 (Ποσοστά πτήσεων Αεροπορικής εταιρείας)

Μια αεροπορική εταιρία ταξιδεύει σε 15 προορισμούς του εσωτερικού. Στα πλαίσια της οικονομικής πολιτικής που πρόκειται να εφαρμόσει, κατέγραψε το ποσοστό πληρότητας των πτήσεων για κάθε μήνα του προηγούμενου ημερολογιακού έτους. Η πολιτική έχει ως εξής:

- Δεν θα γίνει καμία περικοπή σε προορισμούς, στους οποίους το μέσο ετήσιο ποσοστό πληρότητας των πτήσεων είναι μεγαλύτερο του 65.

– Θα γίνουν περικοπές πτήσεων σε προορισμούς, στους οποίους το μέσο ετήσιο ποσοστό πληρότητας των πτήσεων κυμαίνεται από 40 έως και 65. Οι περικοπές θα γίνουν μόνο σε εκείνους τους μήνες που το ποσοστό πληρότητάς τους είναι μικρότερο του 40.

- Θα καταργηθούν οι προορισμοί, στους οποίους το μέσο ετήσιο ποσοστό πληρότητας των πτήσεων είναι μικρότερο του 40.

Να γραφεί αλγόριθμος ο οποίος:

1. Να διαβάζει τα ονόματα των 15 προορισμών και να τα αποθηκεύει σε ένα μονοδιάστατο πίνακα.

2. Να διαβάζει τα ποσοστά πληρότητας των πτήσεων των 15 προορισμών για κάθε μήνα και να τα αποθηκεύει σε δισδιάστατο πίνακα κάνοντας έλεγχο στην καταχώριση των δεδομένων, ώστε να καταχωρούνται μόνο οι τιμές που είναι από 0 έως και 100.

3. Να βρίσκει και να τυπώνει τα ονόματα των προορισμών που δεν θα γίνει καμία περικοπή πτήσεων.

4. Να βρίσκει και να τυπώνει τα ονόματα των προορισμών που θα καταργηθούν.

5. Να βρίσκει και να τυπώνει τα ονόματα των προορισμών, στους οποίους θα γίνουν περικοπές πτήσεων, καθώς και τους μήνες (αύξοντα αριθμό μήνα) που θα γίνουν οι περικοπές.

```
# ----------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ----------------------------------------------------------------
# Όνομα:        2005_1_D
# Περιγραφή:    Ποσοστά πτήσεων Αεροπορικής εταιρείας
# ΕνδΛύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ----------------------------------------------------------------

# 1)
# Εισαγωγή στοιχείων προορισμών
destinations = []
for i in range(15):
    destinations.append(raw_input("Δώστε τον προορισμό = "))

# 2)
# Εισαγωγή στοιχείων πληρότητας
fullness = []
for i in range(15):
    fullness.append([])
    for j in range(12):
        f = (input("Δώστε τη μηνιαία πληρότητα = "))
        while f < 0 or f > 100:
            print "Λάθος δεδομένα ..."
            f = (input("Δώστε τη μηνιαία πληρότητα = "))
        fullness[i].append(f)
```

```
# Υπολογισμός ΜΟ ποσοστού πληρότητας
mo = []
for i in range(15):
    xsum = 0
    for j in range(12):
        xsum += fullness[i][j]
    mo.append(xsum / 12.0)

# 3)
# Προορισμοί χωρίς περικοπή
print "Προορισμοί χωρίς περικοπές ..."
for i in range(15):
    if mo[i] > 65:
        print destinations[i], mo[i]

# 4)
# Προορισμοί που θα ακυρωθούν
print "Προορισμοί που θα ακυρωθούν "
for i in range(15):
    if mo[i] < 40:
        print destinations[i], mo[i]
# 5)
# Προορισμοί με περικοπή
print "Προορισμοί με περικοπές "
for i in range(15):
    if mo[i] >= 40 and mo[i] <= 65:
        print destinations[i], mo[i],
        for j in range(12):
            if fullness[i][j] < 40:
                print " για το μήνα = ", j
```

ΘΕΜΑ Γ_0_2006 (Επιτηρητές σε εξετάσεις ΑΣΕΠ)

Σε ένα διαγωνισμό του ΑΣΕΠ εξετάζονται 1500 υποψήφιοι. Ως εξεταστικό κέντρο χρησιμοποιείται ένα κτίριο με αίθουσες διαφορετικής χωρητικότητας. Ο αριθμός των επιτηρητών που απαιτούνται ανά αίθουσα καθορίζεται αποκλειστικά με βάση τη χωρητικότητα της αίθουσας ως εξής:

ΧΩΡΗΤΙΚΟΤΗΤΑ	ΑΡΙΘΜΟΣ ΕΠΙΤΗΡΗΤΩΝ
Μέχρι και 15 θέσεις	1
Από 16 μέχρι και 23 θέσεις	2
Πάνω από 23 θέσεις	3

Να γίνει πρόγραμμα σε γλώσσα προγραμματισμού «ΓΛΩΣΣΑ» το οποίο:

α. για κάθε αίθουσα θα διαβάζει τη χωρητικότητά της, θα υπολογίζει και θα εμφανίζει τον αριθμό των επιτηρητών που χρειάζονται. Ο υπολογισμός του αριθμού των επιτηρητών να γίνεται από συνάρτηση που θα κατασκευάσετε για το σκοπό αυτό.

β. θα σταματάει όταν εξασφαλισθεί ο απαιτούμενος συνολικός αριθμός θέσεων.

Σημείωση: Να θεωρήσετε ότι η συνολική χωρητικότητα των αιθουσών του κτιρίου επαρκεί για τον αριθμό των υποψηφίων.

```
# ----------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ----------------------------------------------------------------
# Όνομα:       2006_0_C
# Περιγραφή:   Επιτηρητές σε εξετάσεις ΑΣΕΠ
# Ενδλύση:     TasosChatzipapadopoulos
# Python vs:   2.7.10
# Copyright:   (c) 2018
# Url:         http://users.sch.gr/chatzipap
# ----------------------------------------------------------------

# α)
def supervisor(x):
    if x <= 15:
        return 1
    elif x <= 23:
        return 2
    else:
        return 3

# β)
total = 0
total_supervisors = 0
while total < 1500:
    seats = input("Δώσε τις διαθέσιμες θέσεις της αίθουσας = ")
    # πρόβλεψη για την τελαυταία αίθουσα όπου μπορεί να μη γεμίσει
    if 1500 - total < seats:
        s = supervisor(1500 - total)
    else:
        s = supervisor(seats)
    total_supervisors += s
    total += seats
    print 'Επιτηρητές Αίθουσας = ', s, ' Συνολικοί Επιτηρητές = ', total_supervisors
```

ΘΕΜΑ Δ_0_0006 (Καταγραφή θερμοκρασιών επικράτειας)

Για την παρακολούθηση των θερμοκρασιών της επικράτειας κατά το μήνα Μάιο καταγράφεται κάθε μέρα η θερμοκρασία στις 120:00 το μεσημέρι για 200 πόλεις.

Να σχεδιάσετε αλγόριθμο που:

α. θα διαβάζει τα ονόματα των 200 πόλεων και τις αντίστοιχες θερμοκρασίες για κάθε μία από τις ημέρες του μήνα και θα καταχωρεί τα στοιχεία σε πίνακες.

β. θα διαβάζει το όνομα μίας πόλης και θα εμφανίζει τη μέγιστη θερμοκρασία της στη διάρκεια του μήνα. Αν δεν υπάρχει η πόλη στον πίνακα, θα εμφανίζει κατάλληλα διαμορφωμένο μήνυμα.

γ. θα εμφανίζει το πλήθος των ημερών που η μέση θερμοκρασία των 200 πόλεων ξεπέρασε τους 200 oC, αλλά όχι τους 300 oC.

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:        2006_0_D
# Περιγραφή:    Καταγραφή θερμοκρασιών επικράτειας
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ------------------------------------------------------------------

# α)
# Εισαγωγή στοιχείων θερμοκρασιών
names = []
temp = []
for i in range(20):
    names.append(raw_input("Δώσε το όνομα της πόλης = "))
    temp.append([])
    for j in range(30):
        temp[i].append(input("Δώσε τη θερμοκρασία της = "))

# β)
# Εμφάνιση μέσιστης θερμοκρασίας για κάθε πόλη
icity = -1
city = raw_input("Δώσε το όνομα μιας πόλης = ")
for i in range(20):
    if names[i] == city:
        icity = i
if icity != -1:
    tmax = temp[icity][0]
    for j in range(30):
        if temp[icity][j] > tmax:
            tmax = temp[icity][j]
    print names[icity], tmax
else:
    print "Δεν υπάρχει αυτή η πόλη ..."

# γ)
# Εμφάνιση πλήθους ημερών από 20 .. 30
count = 0
for j in range(30):
```

```
    tsum = 0
    for i in range(20):
        tsum += temp[i][j]
    mo = tsum / 20.0
    if mo > 200 and mo <= 30:
        count += 1

print "Πόλεις με θερμοκρασία από [20,30] = ", count
```

ΘΕΜΑ Γ_1_2006 (Πάρκινγκ στάθμευσης αυτοκινήτων)

Σε ένα πάρκινγκ η χρέωση γίνεται κλιμακωτά, όπως φαίνεται στον παρακάτω πίνακα:

ΔΙΑΡΚΕΙΑ ΣΤΑΘΜΕΥΣΗΣ	ΚΟΣΤΟΣ ΑΝΑ ΩΡΑ
Μέχρι και 3 ώρες	2 €
Πάνω από 3 έως και 5 ώρες	1,5 €
Πάνω από 5 ώρες	1,3 €

I. Να κατασκευάσετε πρόγραμμα το οποίο:

α) περιλαμβάνει τμήμα δηλώσεων.

β) για κάθε αυτοκίνητο που στάθμευσε στο πάρκινγκ:

i. διαβάζει τον αριθμό κυκλοφορίας μέχρι να δοθεί το 0. Να θεωρήσετε ότι ο αριθμός κυκλοφορίας μπορεί να περιέχει τόσο γράμματα όσο και αριθμούς.

ii. διαβάζει τη διάρκεια στάθμευσης σε ώρες και τη δέχεται μόνο εφ' όσον είναι μεγαλύτερη από το 0.

iii. καλεί υποπρόγραμμα για τον υπολογισμό του ποσού που πρέπει να πληρώσει ο κάτοχός του.

iv. εμφανίζει τον αριθμό κυκλοφορίας και το ποσό που αναλογεί.

γ) εμφανίζει το πλήθος των αυτοκινήτων που έμειναν στο πάρκινγκ μέχρι και δύο ώρες.

II. Να κατασκευάσετε το υποπρόγραμμα που καλείται στο ερώτημα β) iii.

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:        2006_1_C
# Περιγραφή:    Πάρκινγκ στάθμευσης αυτοκινήτων
# ΕνδΛύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ------------------------------------------------------------------


# γ)
# Υπολογισμός αντιτίμου πάρκινγκ
def pay(t):
    if t <= 3:
        return 2
    elif t <= 5:
        return 1.5
    elif t > 5:
        return 1.3

# β)
plate = raw_input("Δώσε τον αριθμό της πινακίδας = ")
count = 0
while plate != '0':
    t = input("Διάρκεια παραμονής σε ώρες = ")
    while t < 0:
        print "Οι ώρες πρέπει να είναι θετικός αριθμός σε αυτό το σύμπαν ! ! !"
        t = input("Διάρκεια παραμονής σε ώρες = ")
    print "Το όχημα με πινακίδα ", plate, " πρέπει να πληρώσει = ", pay(t) * t
```

```
    if t <= 2:
        count += 1
    plate = raw_input("Δώσε τον αριθμό της πινακίδας = ")

print "Πλήθος αυτοκινήτων με παραμονή κάτω από 2 ώρες = ", count
```

ΘΕΜΑ Δ_1_2006 (Αγώνες Ιππικού Τριάθλου)

Στους προκριματικούς αγώνες ιππικού τριάθλου συμμετέχουν 16 αθλητές. Τα αγωνίσματα είναι: ιππική δεξιοτεχνία, υπερπήδηση εμποδίων και ελεύθερη ιππασία. Ο κάθε αθλητής βαθμολογείται ξεχωριστά σε κάθε ένα από τα τρία αγωνίσματα.

Να σχεδιάσετε αλγόριθμο ο οποίος:

α) καταχωρίζει σε πίνακα τις ονομασίες των τριών αγωνισμάτων, όπως αυτές δίνονται παραπάνω.

β) διαβάζει για κάθε αθλητή όνομα, επίθετο, όνομα αλόγου με το οποίο αγωνίζεται και τους βαθμούς του σε κάθε αγώνισμα και θα καταχωρίζει τα στοιχεία σε πίνακες.

γ) διαβάζει το όνομα και το επίθετο ενός αθλητή και θα εμφανίζει το όνομα του αλόγου με το οποίο αγωνίστηκε και τη συνολική του βαθμολογία στα τρία αγωνίσματα. Αν δεν υπάρχει ο αθλητής, θα εμφανίζει κατάλληλα διαμορφωμένο μήνυμα.

δ) εμφανίζει την ονομασία του αγωνίσματος (ή των αγωνισμάτων) με το μεγαλύτερο «άνοιγμα βαθμολογίας». Ως «άνοιγμα βαθμολογίας» να θεωρήσετε τη διαφορά ανάμεσα στην καλύτερη και στη χειρότερη βαθμολογία του αγωνίσματος.

```
# ----------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ----------------------------------------------------------------
# Όνομα:       2006_1_D
# Περιγραφή:   Αγώνες Ιππικού Τριάθλου
# Ενδλύση:     TasosChatzipapadopoulos
# Python vs:   2.7.10
# Copyright:   (c) 2018
# Url:         http://users.sch.gr/chatzipap
# ----------------------------------------------------------------

# α, β) Εισαγωγή στοιχείων
sports = ["skills", "jumps", "free ride"]
first_names = []
last_names = []
horses = []
grades = []
for i in range(16):
    first_names.append(raw_input("Δώσε το όνομα του αναβάτη = "))
    last_names.append(raw_input('Δώσε το επώνυμο του αναβάτη = '))
    horses.append(raw_input("Δώσε το όνομα του αλόγου = "))
    grades.append([])
    for j in range(3):
        grades[i].append(input("Δώσε τη βαθμολογία στο αγώνισμα = "))

# γ) Εύρεση αναβάτη και εμφάνιση βαθμολογίας
f_name, l_name = raw_input("Δώσε όνομα, επώνυμο αναβάτη= ")
ifound = -1
i = 0
while ifound == -1 and i <= 16:
    if f_name == first_names[i] and l_name == last_names[i]:
        ifound = i
    else:
        ifound += 1
if ifound != -1:
    total = (grades[ifound][0] + grades[ifound][1] + grades[ifound][2]) / 3.0
```

```
    print horses[ifound], total
else:
    search, "Δε βρέθηκε όνομα αναβάτη"

# δ) ονομασία του/των αγωνίσματος/των με το μεγαλύτερο άνοιγμα βαθμολογίας
gap = []
for j in range(3):

    xmax = grades[0][j]
    for i in range(16):
        if grades[i][j] > xmax:
            xmax = grades[i][j]

    xmin = grades[0][j]
    for i in range(16):
        if grades[i][j] < xmin:
            xmin = grades[i][j]

    gap.append(xmax - xmin)

gmax = gap[0]
for j in range(1,3):
    if gap[j] > gmax:
        gmax = gap[j]

for j in range(3):
    if gap[j] == gmax:
        print sports[j]
```

ΘΕΜΑ Γ_0_2007 (Αγοραπωλησίες συλλεκτικών γραμματοσήμων)

Ένας συλλέκτης γραμματοσήμων επισκέπτεται στο διαδίκτυο το αγαπημένο του ηλεκτρονικό κατάστημα φιλοτελισμού προκειμένου να αγοράσει γραμματόσημα. Προτίθεται να ξοδέψει μέχρι 1500 ευρώ.

Να αναπτύξετε αλγόριθμο ο οποίος:

α. Για κάθε γραμματόσημο, να διαβάζει την τιμή και την προέλευσή του (ελληνικό/ξένο) και να επιτρέπει την αγορά του, εφόσον η τιμή του δεν υπερβαίνει το διαθέσιμο υπόλοιπο χρημάτων. Διαφορετικά να τερματίζει τυπώνοντας το μήνυμα «ΤΕΛΟΣ ΑΓΟΡΩΝ».

ΣΗΜΕΙΩΣΗ: Δεν απαιτείται έλεγχος εγκυρότητας για τα δεδομένα εισόδου.

β. Να τυπώνει:

1. Το συνολικό ποσό που ξόδεψε ο συλλέκτης.

2. Το πλήθος των ελληνικών και το πλήθος των ξένων γραμματοσήμων που αγόρασε.

3. Το ποσό που περίσσεψε, εφόσον υπάρχει, διαφορετικά το μήνυμα «ΕΞΑΝΤΛΗΘΗΚΕ ΟΛΟ ΤΟ ΠΟΣΟ».

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:        2007_0_C
# Περιγραφή:    Αγοραπωλησίες συλλεκτικών γραμματοσήμων
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ------------------------------------------------------------------

# α)
total = 0
rest = 1500
greece = abroad = 0
count_greece = count_abroad = 0
end = False
price = input("Δώσε την τιμή του γραμματοσήμου = ")
while not end:
    if price <= rest:
        print "Έγκυρη αγορά ..."
        origin = raw_input("Ποια η προέλευση του γραμματοσήμου (G)reece/(A)broad= ")
        total += price
        rest -= price
        if origin.lower() == "g":
            greece += 1
            count_greece += 1
        elif origin.lower() == "a":
            abroad += 1
            count_abroad += 1
        price = input("Δώσε την τιμη του γραμματοσήμου = ")
    else:
        print "ΤΕΛΟΣ ΑΓΟΡΩΝ "
        end = True

# β)
print total, count_greece, count_abroad
```

```
if rest > 0:
    print rest, " περίσσεψαν"
else:
    print "Δε περίσσεψε φράγκο "
```

ΘΕΜΑ Δ_0_2007 (Πωλήσεις CD δισκογραφικής εταιρείας)

Μια δισκογραφική εταιρεία καταγράφει στοιχεία για ένα έτος για κάθε ένα από τα 20 CDs που κυκλοφόρησε. Τα στοιχεία αυτά είναι ο τίτλος του CD, ο τύπος της μουσικής που περιέχει και οι μηνιαίες του πωλήσεις (ποσά σε ευρώ) στη διάρκεια του έτους. Οι τύποι μουσικής είναι δύο: «ορχηστρική» και «φωνητική».

Να αναπτυχθεί αλγόριθμος ο οποίος:

α. Για κάθε ένα από τα 20 CDs, να διαβάζει τον τίτλο, τον τύπο της μουσικής και τις πωλήσεις του για κάθε μήνα, ελέγχοντας την έγκυρη καταχώριση του τύπου της μουσικής.

β. Να εμφανίζει τον τίτλο ή τους τίτλους των CDs με τις περισσότερες πωλήσεις τον 3ο μήνα του έτους.

γ. Να εμφανίζει τους τίτλους των ορχηστρικών CDs με ετήσιο σύνολο πωλήσεων τουλάχιστον 5000 ευρώ.

δ. Να εμφανίζει πόσα από τα CDs είχαν σύνολο πωλήσεων στο δεύτερο εξάμηνο μεγαλύτερο απ' ό,τι στο πρώτο.

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:        2007_0_D
# Περιγραφή:    Πωλήσεις CD δισκογραφικής εταιρείας
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ------------------------------------------------------------------

# α)
title = []
cd_type = []
sales = []
# Εισαγωγή δεδομένων
for i in range(20):
    title.append(raw_input("Δώσε τον τίτλο του CD = "))
    cd_type.append(raw_input("Δώσε τον τύπο του CD (V)ocal/(O)rchestral ="))
    while cd_type[i].lower() not in ['v', 'o']:
        print "Λάθος τύπος CD ξαναπροσπάθησε ..."
        cd_type[i].append(raw_input("Δώσε τον τύπο του CD (V)ocal/(O)rchestral ="))
    sales.append([])
    for j in range(12):
        sales[i].append(input("Δώσε τις πωλήσεις του CD = "))

# β)
# CD/CDs με τις περισσότερες πωλήσεις
max2 = sales[0][2]
for i in range(1,20):
    if sales[i][2] > max2:
        max2 = sales[i][2]
print "Μεγαλύτερες Πωλήσεις το Μάρτιο"
for i in range(20):
    if sales[i][2] == max2:
        print title[i]

# γ)
```

```
print 'τίτλοι ορχηστρικών CDs με ετήσιο σύνολο πωλήσεων τουλάχιστον 5000 ευρώ.'
for i in range(20):
    xsum = 0
    for j in range(12):
        xsum += sales[i][j]
    if cd_type[i].lower() == 'o' and xsum >= 5000:
        print title[i]

# δ)
print 'πόσα από τα CDs είχαν σύνολο πωλήσεων στο 2ο εξάμηνο μεγαλύτερο του 1ου'
count = 0
for i in range(20):
    sum1 = 0
    for j in range(6):
        sum1 += sales[i][j]
    sum2 = 0
    for j in range(6,12):
        sum2 += sales[i][j]
    if sum2 > sum1:
        count +=1

print count
```

ΘΕΜΑ Γ_1_2007 (Πέτρα - Ψαλίδι - Χαρτί)

Το κλασικό παιχνίδι «Πέτρα-Ψαλίδι-Χαρτί» παίζεται με δύο παίκτες. Σε κάθε γύρο του παιχνιδιού, ο κάθε παίκτης επιλέγει ένα από τα ΠΕΤΡΑ, ΨΑΛΙΔΙ, ΧΑΡΤΙ, και παρουσιάζει την επιλογή του ταυτόχρονα με τον αντίπαλό του. Η ΠΕΤΡΑ κερδίζει το ΨΑΛΙΔΙ, το ΨΑΛΙΔΙ το ΧΑΡΤΙ και το ΧΑΡΤΙ την ΠΕΤΡΑ. Σε περίπτωση που οι δύο παίκτες έχουν την ίδια επιλογή, ο γύρος λήγει ισόπαλος. Το παιχνίδι προχωράει με συνεχόμενους γύρους μέχρι ένας τουλάχιστον από τους παίκτες να αποχωρήσει. Νικητής αναδεικνύεται ο παίκτης με τις περισσότερες νίκες. Αν οι δύο παίκτες έχουν τον ίδιο αριθμό νικών, το παιχνίδι λήγει ισόπαλο.

Να αναπτύξετε αλγόριθμο ο οποίος διαβάζει τα ονόματα των δύο παικτών και υλοποιεί το παραπάνω παιχνίδι ως εξής:

Α. Για κάθε γύρο του παιχνιδιού:

1. διαβάζει την επιλογή κάθε παίκτη, η οποία μπορεί να είναι μία από τις εξής: ΠΕΤΡΑ, ΨΑΛΙΔΙ, ΧΑΡΤΙ, ΤΕΛΟΣ. (Δεν απαιτείται έλεγχος εγκυρότητας τιμών.)

2. συγκρίνει τις επιλογές των παικτών και διαπιστώνει το νικητή του γύρου ή την ισοπαλία.

Β. Τερματίζει το παιχνίδι όταν ένας τουλάχιστον από τους δύο παίκτες επιλέξει ΤΕΛΟΣ.

Γ. Εμφανίζει το όνομα του νικητή ή, αν δεν υπάρχει νικητής, το μήνυμα «ΤΟ ΠΑΙΧΝΙΔΙ ΕΛΗΞΕ ΙΣΟΠΑΛΟ».

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:        2007_1_C
# Περιγραφή:    Πέτρα - Ψαλίδι - Χαρτί
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ------------------------------------------------------------------

# A)
player1 = raw_input("A' Παίκτης. Δώσε το όνομά σου = ")
choice1 = raw_input(str(player1) + " (R)ock, (S)cissors=, (P)aper =")
player2 = raw_input("B' Παίκτης. Δώσε το όνομά σου = ")
choice2 = raw_input(str(player2) + " (R)ock, (S)cissors=, (P)aper = ")
choice1 = choice1.lower()
choice2 = choice2.lower()
wins1 = wins2 = 0

# B), Γ)
while choice1 != 'end' and choice2 != 'end':
    if choice1 == 'r':
        if choice2 == 's':
            wins1 += 1
            print "Winner is", player1
        elif choice2 == 'p':
            wins2 += 1
            print "Winner is", player2
        else:
            print "DRAW"
    elif choice1 == 's':
        if choice2 == 'p':
```

```
            wins1 += 1
            print "Winner is", player1
        elif choice2 == 'r':
            wins2 += 1
            print "Winner is", player2
        else:
            print "DRAW"
    elif choice1 == 'p':
        if choice2 == 'r':
            wins1 += 1
            print "Winner is", player1
        elif choice2 == 's':
            wins2 += 1
            print "Winner is", player2
        else:
            print "DRAW"
    choice1 = raw_input(str(player1) + " (R)ock, (S)cissors=, (P)aper =")
    choice2 = raw_input(str(player2) + " (R)ock, (S)cissors=, (P)aper = ")
    choice1 = choice1.lower()
    choice2 = choice2.lower()

print "---------ΑΠΟΤΕΛΕΣΜΑΤΑ----------"
print player1, player2
print wins1, wins2
if wins1 > wins2:
    print "WINNER IS ...", player1
elif wins2 > wins1:
    print "WINNER IS ...", player2
else:
    print "It's a DRAW"
```

ΘΕΜΑ Δ_1_2007 (Παραγωγή αυγών Πτηνοτροφικής μονάδας)

Μια σύγχρονη πτηνοτροφική μονάδα παρακολουθεί την ημερήσια παραγωγή αυγών και καταγράφει τα στοιχεία σε ηλεκτρονικό αρχείο. Να αναπτύξετε αλγόριθμο ο οποίος θα

διαχειρίζεται τα στοιχεία της μονάδας στη διάρκεια ενός έτους. Για το σκοπό αυτό:

Α. Να κατασκευάσετε κύριο πρόγραμμα το οποίο:

1. να ζητάει το έτος παρακολούθησης, ελέγχοντας ότι πρόκειται για έτος του 21ου αιώνα (από 2000 μέχρι και 2099). Ο αλγόριθμος να δημιουργεί πίνακα με τον αριθμό των ημερών για καθέναν από τους δώδεκα μήνες του έτους που δόθηκε. Ο αριθμός των ημερών του μήνα θα υπολογίζεται από υποπρόγραμμα το οποίο θα κατασκευάσετε για το σκοπό αυτό. Η λειτουργία του υποπρογράμματος περιγράφεται στο ερώτημα Β.

2. να ζητάει την ημερήσια παραγωγή (αριθμό αυγών) για κάθε μέρα του έτους και να καταχωρίζει τις τιμές σε πίνακα δύο διαστάσεων, με μια γραμμή για κάθε μήνα.

3. να εμφανίζει τον τρίτο κατά σειρά από τους μήνες του έτους που έχουν ο καθένας μέσο όρο ημερήσιας παραγωγής μέχρι και δέκα ποσοστιαίες μονάδες πάνω ή κάτω από τον ετήσιο μέσο όρο. Αν δεν βρει τέτοιο μήνα, να εμφανίζει κατάλληλο μήνυμα.

Β. Να κατασκευάσετε υποπρόγραμμα το οποίο να δέχεται ως παραμέτρους κάποιο έτος και τον αριθμό κάποιου μήνα (1 έως 12), και να επιστρέφει τον αριθμό των ημερών του συγκεκριμένου μήνα. Όταν το έτος είναι δίσεκτο, ο Φεβρουάριος έχει 29 ημέρες, διαφορετικά έχει 28. Δίσεκτα είναι τα έτη που διαιρούνται με το 4 αλλά όχι με το 100, καθώς και εκείνα που διαιρούνται με το 400. Για τους υπόλοιπους μήνες, πλην του Φεβρουαρίου, ισχύει το εξής: μέχρι και τον Ιούλιο (7ο ς μήνας) οι μονοί μήνες έχουν 31 ημέρες και οι ζυγοί 30. Για τους μήνες μετά τον Ιούλιο, ισχύει το αντίστροφο.

```
# ----------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ----------------------------------------------------------------
# Όνομα:       2007_1_D
# Περιγραφή:   Παραγωγή αυγών Πτηνοτροφικής μονάδας
# ΕνδΛύση:     TasosChatzipapadopoulos
# Python vs:   2.7.10
# Copyright:   (c) 2018
# Url:         http://users.sch.gr/chatzipap
# ----------------------------------------------------------------

# 3.B)
def month_days(year, month):
    leap = False
    if (year % 4 == 0 and year % 100 != 0) or year % 400 == 0:
        leap = True
    if month in [1, 3, 5, 7]:
        days = 31
    elif month in [4, 6]:
        days = 30
    elif month in [8, 10, 12]:
        days = 31
    elif month in [9, 11]:
        days = 30
    elif month == 2:
        if leap == False:
            days = 28
```

```
        else:
            days = 29
    return days

# 1)
# Εισαγωγή στοιχείων (1)
year = input("Εισαγωγή έτους = ")
while year < 2000 or year > 2099:
    print "¨Ετος έξω από τα αποδεκτά όρια (2000..2099)"
    year = input("Εισαγωγή έτους = ")
days_per_month = []
for month in range(1, 13, 1):
    days_per_month.append(month_days(year, month))
eggs = []

# 2)
# Εισαγωγή στοιχείων (2)
from random import randint
for month in range(12):
    # Για να έχουμε στοιχεία δοκιμής προτιμούμε
    # αυτόματη εισαγωγή τυχαίου αριθμού αυγών από 0..9
    eggs.append([])
    for day in range(days_per_month[month]):
        eggs[month].append(randint(0, 9))

# 3)
# Υπολογισμός ερωτηματος (3)
# Υπολογισμός μέσης ετήσιας παραγωγής
xsum = 0
year_days = 0
for month in range(12):
    for day in range(days_per_month[month]):
        xsum += eggs[month][day]
        year_days += 1
year_average = float(xsum) / float(year_days)
print "Μέσος όρος έτους = ", year_average
count = 0
for month in range(12):
    month_sum = 0
    for day in range(days_per_month[month]):
        month_sum += eggs[month][day]
    month_average = float(month_sum) / float(days_per_month[month])
    difference = abs(month_average - year_average) * 100 / year_average
    print difference,
    if difference <= 10:
        count += 1
    if count == 3:
        print "Ο 3ος μήνας υπάρχει και είναι ο  ", month+1, " στη σειρά"
        break
if count != 3:
    print "Δεν υπάρχουν 3 στη σειρά μήνες με διαφορά 10% "
```

ΘΕΜΑ Γ_0_2008 (Ενοικίαση Αυτοκινήτων)

Μία εταιρεία ενοικίασης αυτοκινήτων έχει νοικιάσει 30 αυτοκίνητα τα οποία κατηγοριοποιούνται σε οικολογικά και συμβατικά. Η πολιτική χρέωσης για την ενοικίαση ανά κατηγορία και ανά ημέρα δίνεται στον παρακάτω πίνακα.

ΗΜΕΡΕΣ	ΟΙΚΟΛΟΓΙΚΑ	ΣΥΜΒΑΤΙΚΑ
1-7	30€ ανά ημέρα	40€ ανά ημέρα
8-16	20€ ανά ημέρα	30€ ανά ημέρα
από 17 και άνω	10€ ανά ημέρα	20€ ανά ημέρα

1. Να αναπτύξετε πρόγραμμα το οποίο:

α. Περιλαμβάνει τμήμα δηλώσεων μεταβλητών.

β. Για κάθε αυτοκίνητο το οποίο έχει ενοικιαστεί:

i. Διαβάζει την κατηγορία του («ΟΙΚΟΛΟΓΙΚΑ» ή «ΣΥΜΒΑΤΙΚΑ») και τις ημέρες ενοικίασης.

ii. Καλεί υποπρόγραμμα με είσοδο την κατηγορία του αυτοκινήτου και τις ημέρες ενοικίασης και υπολογίζει με βάση τον παραπάνω πίνακα τη χρέωση.

iii. Εμφανίζει το μήνυμα “χρέωση” και τη χρέωση που υπολογίσατε.

γ. Υπολογίζει και εμφανίζει το πλήθος των οικολογικών και των συμβατικών αυτοκινήτων.

2. Να κατασκευάσετε το κατάλληλο υποπρόγραμμα του ερωτήματος 1.β.ii .

ΣΗΜΕΙΩΣΗ: 1)Δεν απαιτείται έλεγχος εγκυρότητας για τα δεδομένα εισόδου και 2) Ο υπολογισμός της χρέωσης δεν πρέπει να γίνει κλιμακωτά.

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:        2008_0_C
# Περιγραφή:    Ενοικίαση Αυτοκινήτων
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ------------------------------------------------------------------


# 2)
def charge(category, days):
    if category.upper() == "E":
        if days <= 7:
            money = days * 30
        elif days <= 16:
            money = days * 20
        else:
            money = days * 10
    if category.upper() == "T":
        if days <= 7:
            money = days * 40
        elif days <= 16:
            money = days * 30
        else:
            money = days * 20
```

```
    return money

# 1β)
eco = typical = 0
for car in range(30):
    category = raw_input("Δώσε την κατηγορία του αυτοκινήτου (E)co/(T)ypical = ")
    days = input("Πόσες ημέρες ενοικίασης = ")
    if category.upper() == "E":
        eco += 1
    if category.upper() == "T":
        typical += 1

    print "Χρέωση = ", charge(category, days)

# γ)
print "Οικολογικά Αυτοκίνητα = ", eco
print "Κανονικά Αυτοκίνητα = ", typical
```

ΘΕΜΑ Δ_0_2008 (Αποτελέσματα ευρωπαϊκού πρωταθλήματος ποδοσφαίρου)

Στο ευρωπαϊκό πρωτάθλημα ποδοσφαίρου συμμετέχουν 16 ομάδες. Κάθε ομάδα συμμετέχει σε 30 αγώνες. Να γράψετε αλγόριθμο ο οποίος:

α. Διαβάζει σε μονοδιάστατο πίνακα ΟΝ[16] τα ονόματα των ομάδων.

β. Διαβάζει σε δισδιάστατο πίνακα ΑΠ[16,30] τα αποτελέσματα σε κάθε αγώνα ως εξής:

Τον χαρακτήρα «Ν»για ΝΙΚΗ

Τον χαρακτήρα «Ι» για ΙΣΟΠΑΛΙΑ

Τον χαρακτήρα «Η» για ΗΤΤΑ

και κάνει τον απαραίτητο έλεγχο εγκυρότητας των δεδομένων.

γ. Για κάθε ομάδα υπολογίζει και καταχωρεί σε δισδιάστατο πίνακα ΠΛ[16,3] το πλήθος των νικών στην πρώτη στήλη, το πλήθος των ισοπαλιών στη δεύτερη στήλη, και το πλήθος των ηττών στην τρίτη στήλη του πίνακα. Ο πίνακας αυτός πρέπει προηγουμένως να έχει μηδενισθεί.

δ. Με βάση τα στοιχεία του πίνακα ΠΛ[16,3] υπολογίζει και καταχωρεί σε νέο πίνακα ΒΑΘ[16] τη συνολική βαθμολογία κάθε ομάδας, δεδομένου ότι για κάθε νίκη η ομάδα παίρνει τρεις βαθμούς, για κάθε ισοπαλία έναν βαθμό και για κάθε ήττα κανέναν βαθμό.

ε. Εμφανίζει τα ονόματα και τη βαθμολογία των ομάδων ταξινομημένα σε φθίνουσα σειρά με βάση τη βαθμολογία.

```
# ---------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ---------------------------------------------------------------
# Όνομα:        2008_0_D
# Περιγραφή:    Αποτελέσματα ευρωπαϊκού πρωταθλήματος ποδοσφαίρου
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ---------------------------------------------------------------

# Εισαγωγή Δεδομένων α), β)
teams = []
for i in range(16):
    teams.append(raw_input("Δώσε το όνομα της ομάδας = "))

results = []
for team in range(16):
    results.append([])
    for game in range(30):
        a = raw_input("Δώσε το αποτέλεσμα (w)win/(d)draw/(f)defeat = ")
        while a.lower() not in ['w','d','f']:
            print "Λάθος δεδομένα ..."
            a = raw_input("Δώσε το αποτέλεσμα (w)win/(d)draw/(f)defeat = ")
        results[team].append(raw_input("Δώσε το αποτέλεσμα (w)win/(d)draw/(f)defeat =
"))

# Ερώτημα γ)
count = [[0,0,0] for i in range(16)]
for i in range(16):
    w = d = f = 0
```

```
    for j in range(30):
        if results[i][j] == 'w':
            count[i][0] += 1
        elif results[i][j] == 'd':
            count[i][1] += 1
        elif results[i][j] == 'f':
            count[i][2] += 1

# Ερώτημα δ)
scores = []
for i in range(16):
    scores[i].append(count[i][0] * 3 + count[i][1])

# Ερώτημα ε)
for i in range(15):
    for j in range(15, i, -1):
        if scores[j] > scores[j-1]:
            scores[j], scores[j-1] = scores[j-1], scores[j]
            teams[j], teams[j-1] = teams[j-1], teams[j]

print '------Αποτελέσματα-----'
for i in range(16):
    print teams[i], scores[i]
```

ΘΕΜΑ Γ_1_2008 (Βοηθητικο επίδομα μισθοδοσίας)

Μία εταιρεία αποφάσισε να δώσει βοηθητικό επίδομα στους υπαλλήλους της για τον μήνα Ιούλιο. Το επίδομα διαφοροποιείται, ανάλογα με το φύλο του/της υπαλλήλου και τον αριθμό των παιδιών του/της, με βάση τους παρακάτω πίνακες:

ΑΝΔΡΕΣ	
ΑΡΙΘΜΟΣ ΠΑΙΔΙΩΝ	ΕΠΙΔΟΜΑ ΣΕ €
1	20
2	50
>=3	120
ΓΥΝΑΙΚΕΣ	
ΑΡΙΘΜΟΣ ΠΑΙΔΙΩΝ	ΕΠΙΔΟΜΑ ΣΕ €
1	30
2	80
>= 3	160

Να γράψετε αλγόριθμο ο οποίος

α. διαβάζει το φύλο («Α» ή «Γ») το οποίο ελέγχεται ως προς την ορθότητα της εισαγωγής του. Επίσης διαβάζει τον μισθό και τον αριθμό των παιδιών του υπαλλήλου.

Μονάδες 3

β. υπολογίζει και εμφανίζει το επίδομα και το συνολικό ποσό που θα εισπράξει ο υπάλληλος τον μήνα Ιούλιο.

Μονάδες 7

γ. δέχεται απάντηση «ΝΑΙ» ή «ΟΧΙ» για τη συνέχεια ή τον τερματισμό της επανάληψης μετά την εμφάνιση σχετικού μηνύματος.

Μονάδες 4

δ. υπολογίζει και εμφανίζει το συνολικό ποσό επιδόματος που πρέπει να καταβάλει η Εταιρεία στους υπαλλήλους της.

Μονάδες 6

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:        2008_1_C
# Περιγραφή:    Βοηθητικο επίδομα μισθοδοσίας
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ------------------------------------------------------------------

total = 0
e = raw_input('Θέλεις να υπολογίσεις επίδομα (Y)es, (N)o = ')
while e.upper() != 'N':
    # α)
    gender = raw_input("Δώσε το φύλο (M)ale, (F)emale = ")
    while gender.upper() not in ['M', 'F']:
```

```
        print 'Λάθος δεδομένα ...'
        gender = raw_input("Δώσε το φύλο (M)ale, (F)emale = ")

    salary = input("Δώσε το μισθό του/της υπαλλήλου = ")
    kids = input("Δώσε το πλήθος των παιδιών = ")

    # β)
    if gender == 'M':
        if kids == 1:
            allowance = 20
        elif kids == 2:
            allowance = 50
        else:
            allowance = 120
    else:
        if kids == 1:
            allowance = 30
        elif kids == 2:
            allowance = 80
        else:
            allowance = 160
    print ' Επίδομα / Μισθός'
    print allowance, salary

    total += allowance
# γ)
    e = raw_input('Θέλεις να υπολογίσεις επίδομα (Y)es, (N)o = ')

print 'Συνολικό επίδομα που πρέπει να καταβληθεί στους υπαλλήλους = ', total
```

ΘΕΜΑ Δ_1_2008 (Αποτελέσματα 110μ με εμπόδια)

Στο άθλημα των 110 μέτρων μετ' εμποδίων, στους δύο ημιτελικούς αγώνες συμμετέχουν δέκα έξι (16) αθλητές (8 σε κάθε ημιτελικό). Σύμφωνα με τον κανονισμό στον τελικό προκρίνεται ο πρώτος αθλητής κάθε ημιτελικού. Η οκτάδα του τελικού συμπληρώνεται με τους αθλητές που έχουν τους έξι (6) καλύτερους χρόνους απ' όλους τους υπόλοιπους συμμετέχοντες. Να θεωρήσετε ότι δεν υπάρχουν αθλητές με ίδιους χρόνους.

1. Να γράψετε πρόγραμμα στη «ΓΛΩΣΣΑ» το οποίο

α. περιλαμβάνει το τμήμα δηλώσεων.

β. καλεί τη διαδικασία ΕΙΣΟΔΟΣ για κάθε ημιτελικό ξεχωριστά. Η διαδικασία διαβάζει το όνομα του αθλητή και το χρόνο του (με ακρίβεια δεκάτου του δευτερολέπτου).

γ. καλεί τη διαδικασία ΤΑΞΙΝΟΜΗΣΗ για κάθε ημιτελικό ξεχωριστά. Η διαδικασία ταξινομεί τους αθλητές ως προς τον χρόνο τους με αύξουσα σειρά.

δ. δημιουργεί τον πίνακα ΟΝ με τα ονόματα και τον πίνακα ΧΡ με τους αντίστοιχους χρόνους των αθλητών που προκρίθηκαν στον τελικό.

ε. εμφανίζει τα ονόματα και τους χρόνους των αθλητών που θα λάβουν μέρος στον τελικό.

2. Να γράψετε

α. τη διαδικασία ΕΙΣΟΔΟΣ.

β. τη διαδικασία ΤΑΞΙΝΟΜΗΣΗ.

```
# ---------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ---------------------------------------------------------------
# Όνομα:       2008_1_D
# Περιγραφή:   Αποτελέσματα 110μ με εμπόδια
# Ενδλύση:     TasosChatzipapadopoulos
# Python vs:   2.7.10
# Copyright:   (c) 2018
# Url:         http://users.sch.gr/chatzipap
# ---------------------------------------------------------------

# Ερώτημα α), β)
def read():
    names = []
    times = []
    for i in range(8):
        names.append(raw_input('Δώσε το όνομα του αθλητή = '))
        time = input('Δώσε το χρόνο του αθλητή = ')
        times.append(round(time, 1))
    return names, times

names1, times1 = read()
names2, times2 = read()


# Ερώτημα γ)
def my_sort(names, times):
    for i in range(7):
        for j in range(7, i, -1):
            if times[j] < times[j-1]:
```

```
                times[j], times[j-1] = times[j-1], times[j]
                names[j], names[j-1] = names[j-1], names[j]
# Ερώτημα δ)
my_sort(names1, times1)
my_sort(names2, times2)

onoma = [0] * 8
xronos = [0] * 8

onoma[0] = names1[0]
xronos[0] = times1[0]
onoma[1] = names2[1]
xronos[1] = times2[1]
m = n = 2

for i in range(2, 8):
    if times1[i] < times2[i]:
        onoma[i] = names1[m]
        xronos[i] = times1[m]
        m += 1
    else:
        onoma[i] = names2[n]
        xronos[i] = times2[n]
        n += 1

# ε)
my_sort(onoma, xronos)
print "Οι αθλητές που προκρίθηκαν στο τελικό"
for i in range(8):
    print onoma[i], xronos[i]
```

ΘΕΜΑ Γ_0_2009 (Επιβίβαση-Αποβίβαση Επιβατών τρένου)

Σε μια διαδρομή τρένου υπάρχουν 20 σταθμοί (σε αυτούς περιλαμβάνονται η αφετηρία και ο τερματικός σταθμός). Το τρένο σταματά σε όλους τους σταθμούς. Σε κάθε σταθμό επιβιβάζονται και αποβιβάζονται επιβάτες. Οι πρώτοι επιβάτες επιβιβάζονται στην αφετηρία και στον τερματικό σταθμό αποβιβάζονται όλοι οι επιβάτες.

Να κατασκευάσετε αλγόριθμο, ο οποίος να διαχειρίζεται την κίνηση των επιβατών. Συγκεκριμένα:

Α. Να ζητάει από το χρήστη τον αριθμό των ατόμων που επιβιβάστηκαν σε κάθε σταθμό, εκτός από τον τερματικό, και να τον εισάγει σε πίνακα ΕΠΙΒ[19].

Β. Να εισάγει σε πίνακα ΑΠΟΒ[19] τον αριθμό των ατόμων που αποβιβάστηκαν σε κάθε σταθμό, εκτός από τον τερματικό, ως εξής:

Για την αφετηρία να εισάγει την τιμή μηδέν (0) και για τους υπόλοιπους σταθμούς να ζητάει από τον χρήστη τον αριθμό των ατόμων που αποβιβάστηκαν.

Γ. Να δημιουργεί πίνακα ΑΕ[19], στον οποίο να καταχωρίζει τον αριθμό των επιβατών που βρίσκονται στο τρένο, μετά από κάθε αναχώρησή του.

Δ. Να βρίσκει και να εμφανίζει τον σταθμό από τον οποίο το τρένο αναχωρεί με τον μεγαλύτερο αριθμό επιβατών.

(Να θεωρήσετε ότι από κάθε σταθμό το τρένο αναχωρεί με διαφορετικό αριθμό επιβατών).

```
# -------------------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# -------------------------------------------------------------------------------
# Όνομα:       2009_0_C
# Περιγραφή:   Επιβίβαση-Αποβίβαση Επιβατών τρένου
# Λύση:        Anastasios Chatzipapadopoulos
# Created:     2017
# Python vs:   2.7.10
# Copyright:   (c) 2018
# http://      users.sch.gr/chatzipap
# -------------------------------------------------------------------------------

# Ερώτημα Α
on_board = []
for i in range(19):
    on_board.append(input('Πόσοι επιβάτες Επιβιβάστηκαν στον σταθμό ' + str(i) +' = '))
on_board.append(0)

# Ερώτημα Β
off_board = []
off_board.append(0)
for i in range(1, 20):
    off_board.append(input('Πόσοι επιβάτες Αποβιβάστηκαν στον σταθμό ' + str(i) +' =
'))

# Ερώτημα Γ
current_passengers = []
total = 0
for i in range(20):
  total += on_board[i] - off_board[i]
  current_passengers.append(total)
```

```
# Ερώτημα Δ
xmax = current_passengers[0]
ixmax = 0
for i in range(1, 20):
    if current_passengers[i] > xmax:
        xmax = current_passengers[i]
        ixmax = i

print 'Σταθμός με μεγαλύτερο αριθμό επιβατών / επιβάτες'
print ixmax, xmax
```

ΘΕΜΑ Δ_0_2009 (Μίσθωση δωματίων Ξενοδοχείου)

Ξενοδοχειακή επιχείρηση διαθέτει 25 δωμάτια. Τα δωμάτια αριθμούνται από το 1 μέχρι το 25. Ο συνολικός αριθμός των υπαλλήλων που απασχολούνται ημερησίως στο ξενοδοχείο εξαρτάται από τα κατειλημμένα δωμάτια και δίνεται από τον παρακάτω πίνακα

Αριθμός κατειλημμένων δωματίων	Συνολικός αριθμός υπαλλήλων
από 0 μέχρι 4	= 3
από 5 μέχρι 8	= 4
από 9 μέχρι 12	= 5
πάνω από 12	= 6

Η ημερήσια χρέωση για κάθε δωμάτιο είναι 75€ και το ημερομίσθιο κάθε υπαλλήλου 45€.

Α. Να κατασκευάσετε κύριο πρόγραμμα το οποίο:

1. Να περιλαμβάνει τμήμα δηλώσεων.

2. Να διαβάζει σε πίνακα ΚΡΑΤ[25,7] την κατάσταση κάθε δωματίου για κάθε μέρα της εβδομάδας, ελέγχοντας την ορθή καταχώριση. Το πρόγραμμα να δέχεται μόνο τους χαρακτήρες «Κ» για κατειλημμένο, «Δ» για διαθέσιμο αντίστοιχα.

3. Να υπολογίζει το συνολικό κέρδος ή τη συνολική ζημιά κατά τη διάρκεια της εβδομάδας και να εμφανίζει κατάλληλο μήνυμα. Για το σκοπό αυτό να καλεί το υποπρόγραμμα ΚΕΡΔΟΣ, που περιγράφεται στο ερώτημα Β.

Β. Να αναπτύξετε το υποπρόγραμμα ΚΕΡΔΟΣ, το οποίο να δέχεται τον πίνακα των κρατήσεων και έναν αριθμό ημέρας (από 1 έως 7). Το υποπρόγραμμα να υπολογίζει και να επιστρέφει το κέρδος της συγκεκριμένης ημέρας. Το κέρδος κάθε ημέρας προκύπτει από τα ημερήσια έσοδα ενοικιάσεων, αν αφαιρεθούν τα ημερομίσθια των υπαλλήλων της συγκεκριμένης ημέρας. Αν τα έσοδα είναι μικρότερα από τα ημερομίσθια, το κέρδος είναι αρνητικό (ζημιά).

```
# ----------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ----------------------------------------------------------------
# Όνομα:        2009_0_D
# Περιγραφή:    Μίσθωση δωματίων Ξενοδοχείου
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ----------------------------------------------------------------

# Ερώτημα 1, 2
booking = []
for room in range(25):
    booking.append([])
    for day in range(7):
        state = raw_input('Δώσε την κατάσταση του δωματίου (A)Available/(B)Booked =
')
        while state.upper() not in ['A', 'B']:
            print 'Λάθος δεδομένα ξαναπροσπάθησε...'
            state = raw_input('Δώσε την κατάσταση του δωματίου (A)Available/(B)Booked
= ')
        booking[room].append(state.upper())
```

```
# Ερώτημα Β
def profit(booking, day):
    if day in range(7):
        count = 0
        for room in range(25):
            if booking[room][day] == 'B':
                count += 1
        gain = count * 75
        if count <= 4:
            gain -= 3 * 45
        elif count <= 8:
            gain -= 4 * 45
        elif count <= 12:
            gain -= 5 * 45
        else:
            gain -= 6 * 45
        return gain
    else:
        raise ValueError('Μόνο 7 ημέρες η εβδομάδα παρακαλώ ...')

# Ερώτημα 3
total = 0
for day in range(7):
    total += profit(booking, day)
if total > 0:
    print 'ΚΕΡΔΟΣ = ',
elif total < 0:
    print 'ΖΗΜΙΑ = ',
else:
    print 'ΜΗΔΕΝΙΚΟ ΚΕΡΔΟΣ = ',
print total
```

ΘΕΜΑ Γ_1_2009 (Αναβαθμολόγηση Γραπτών Εξετάσεων)

Στις γενικές εξετάσεις, κάθε γραπτό βαθμολογείται από δύο βαθμολογητές στην κλίμακα 1-100. Όταν η διαφορά των δύο βαθμών είναι μεγαλύτερη από δώδεκα μονάδες, το γραπτό αναβαθμολογείται, δηλαδή βαθμολογείται και από τρίτο βαθμολογητή. Στα γραπτά που δεν έχουν αναβαθμολογηθεί, ο τελικός βαθμός προκύπτει από το πηλίκο της διαίρεσης του αθροίσματος των βαθμών των δύο βαθμολογητών διά δέκα. Στα γραπτά που έχουν αναβαθμολογηθεί, ο τελικός βαθμός προκύπτει με τον ίδιο τρόπο, αλλά λαμβάνονται υπόψη οι δύο μεγαλύτεροι βαθμοί. Για στατιστικούς λόγους, οι τελικοί βαθμοί (ΤΒ) κατανέμονται στις παρακάτω βαθμολογικές κατηγορίες:

1η	2η	3η	4η	5η	6η
0≤ΤΒ<5	5≤ΤΒ<10	10≤ ΤΒ<12	12≤ΤΒ<15	15≤ΤΒ<18	18≤ΤΒ≤20

Σ' ένα βαθμολογικό κέντρο υπάρχουν 780 γραπτά στο μάθημα «Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον». Οι βαθμοί των δύο βαθμολογητών έχουν καταχωριστεί στις δύο πρώτες στήλες ενός πίνακα Β[780,3].

Να γραφεί αλγόριθμος ο οποίος:

Α. Να ελέγχει, για κάθε γραπτό, αν χρειάζεται αναβαθμολόγηση. Αν χρειάζεται, να ζητάει από τον χρήστη τον βαθμό του τρίτου βαθμολογητή και να τον εισάγει στην αντίστοιχη θέση της τρίτης στήλης, διαφορετικά να εισάγει την τιμή -1. Δεν απαιτείται έλεγχος εγκυρότητας.

Β. Να υπολογίζει τον τελικό βαθμό κάθε γραπτού και να τον καταχωρίζει στην αντίστοιχη θέση ενός πίνακα Τ[780].

Γ. Να εμφανίζει τη βαθμολογική κατηγορία (ή τις κατηγορίες) με το μεγαλύτερο πλήθος γραπτών.

```
# ---------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ---------------------------------------------------------------------
# Όνομα:        2009_1_C
# Περιγραφή:    Αναβαθμολόγηση Γραπτών Εξετάσεων
# ΕνδΛύση:      TasosChatzipapadopoulos
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ---------------------------------------------------------------------

# Τυχαιοποίηση Λίστας Βαθμολογιών
from random import randint
grades = []
for i in range(780):
    grades.append([randint(1, 100), randint(1, 100), None])

# Ερώτημα Α
for i in range(780):
    diaf = grades[i][0] - grades[i][1]
    if diaf < 0:
        diaf *= -1
    if diaf > 12:
        grades[i][2] = randint(1, 100)
    else:
        grades[i][2] = -1
```

```
def max2(a, b, c):
    if a >= b and b >= c:
        return [a, b]
    elif b >= c and c >= a:
        return [b, c]
    else:
        return [c, a]

# Ερώτημα Β
final = []
for i in range(780):
    if grades[i][2] == -1:
        final.append((grades[i][0] + grades[i][1]) / 10.0)
    else:
        a, b = max2(grades[i][0], grades[i][1], grades[i][2])
        final.append((a + b) / 10.0)

# Ερώτημα Γ
count = 7 * [0]
for i in range(780):
    if final[i] < 5:
        count[0] += 1
    elif final[i] < 10:
        count[1] += 1
    elif final[i] < 12:
        count[2] += 1
    elif final[i] < 15:
        count[3] += 1
    elif final[i] < 18:
        count[4] += 1
    else:
        count[5] += 1

xmax = count[0]
for i in range(6):
    if count[i] > xmax:
        xmax = count[i]

print "Μεγαλύτερο πλήθος γραπτών είχαν οι κατηγορίες ..."
for i in range(6):
    if xmax == count[i]:
        print i,
```

ΘΕΜΑ Δ_1_2009 (Tic Tac Toe Game)

Το παιχνίδι τρίλιζα παίζεται με διαδοχικές κινήσεις δύο παικτών σε έναν πίνακα Τ[3,3]. Οι παίκτες συμπληρώνουν εναλλάξ μια θέση του πίνακα, τοποθετώντας ο μεν πρώτος το σύμβολο-χαρακτήρα 'Χ', ο δε δεύτερος το σύμβολο– χαρακτήρα 'Ο'. Νικητής είναι ο παίκτης που θα συμπληρώσει πρώτος μια τριάδα όμοιων συμβόλων σε κάποια γραμμή, στήλη ή διαγώνιο του πίνακα. Αν ο πίνακας συμπληρωθεί χωρίς νικητή, το παιχνίδι θεωρείται ισόπαλο.

Α. Να γράψετε πρόγραμμα στη «ΓΛΩΣΣΑ», το οποίο:

1. Να τοποθετεί σε κάθε θέση του πίνακα Τ τον χαρακτήρα '-'.

2. Για κάθε κίνηση:

α. Να δέχεται τις συντεταγμένες μιας θέσης του πίνακα Τ και να τοποθετεί στην αντίστοιχη θέση το σύμβολο του παίκτη. Να θεωρήσετε ότι οι τιμές των συντεταγμένων είναι πάντοτε σωστές (1 έως 3) είναι όμως αποδεκτές, μόνον αν η θέση που προσδιορίζουν δεν περιέχει ήδη ένα σύμβολο παίκτη.

β. Να ελέγχει εάν με την κίνησή του ο παίκτης νίκησε. Για τον σκοπό αυτόν, να καλεί τη συνάρτηση ΝΙΚΗΣΕ, που περιγράφεται στο ερώτημα Β.

3. Να τερματίζει το παιχνίδι, εφόσον σημειωθεί ισοπαλία ή νικήσει ένας από τους δύο παίκτες.

4. Να εμφανίζει με κατάλληλο μήνυμα (πρώτος παίκτης/ δεύτερος παίκτης/ισοπαλία) το αποτέλεσμα του παιχνιδιού.

Β. Να κατασκευάσετε τη συνάρτηση ΝΙΚΗΣΕ, η οποία θα δέχεται τον πίνακα Τ και τις συντεταγμένες (Γ, Σ) μιας θέσης του πίνακα και θα επιστρέφει την τιμή ΑΛΗΘΗΣ, αν υπάρχει τρεις φορές το ίδιο σύμβολο, σε τουλάχιστον μια από τις παρακάτω περιπτώσεις:

1. Στη γραμμή Γ.

2. Στη στήλη Σ.

3. Στην κύρια διαγώνιο (δηλαδή Γ=Σ).

4. Στη δευτερεύουσα διαγώνιο (δηλαδή Γ+Σ=4).

Σε κάθε άλλη περίπτωση, η συνάρτηση να επιστρέφει την τιμή ΨΕΥΔΗΣ.

```
# ---------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ---------------------------------------------------------------
# Όνομα:       2009_1_D
# Περιγραφή:   Tic Tac Toe Game
# ΕνδΛύση:     TasosChatzipapadopoulos
# Python vs:   2.7.10
# Copyright:   (c) 2018
# Url:         http://users.sch.gr/chatzipap
# ---------------------------------------------------------------

# Η εκφώνηση θέλει συμπλήρωμα
# Η συνάρτηση πρέπει να έχει και 3ο όρισμα το σύμβολο που ψάχνουμε
# καθόσον στην αρχή έχουν όλες οι θέσεις τον ίδιο χαρακτήρα '-'
# Ερώτημα 1
t_list = [['-'] * 3 for i in range(3)]


# Ερώτημα Β
def wins(t, x, y, s):
    x -= 1
```

```
    y -= 1
    if t[x][0] == t[x][1] == t[x][2] == s or \
       t[0][y] == t[1][y] == t[2][y] == s or \
       t[0][0] == t[1][1] == t[2][2] == s or \
       t[0][2] == t[1][1] == t[2][0] == s:
        return True
    else:
        return False


# Εμφάνιση της τρίλιζας
def show():
    for item in t_list:
        print item

# Ερώτημα 2
turns = 0
win = False
while turns < 9 and win == False:
    show()
    # ¨Ελεγχος δεδομένων
    x, y = input('Δώσε συντεταγμένες x, y = ')
    while t_list[x-1][y-1] != '-':
        print '%d, %d η θέση είναι κατειλημμένη' % (x, y)
        x, y = input('Δώσε συντεταγμένες x, y = ')
    turns += 1
    if turns % 2 == 0:
        s = 'X'
    else:
        s = 'O'
    print s, 'Σύμβολο καταχωρήθηκε'
    t_list[x-1][y-1] = s
    if wins(t_list, x, y, s):
        print "Νικητής ο παίκτης με το σύμβολο ", s
        win = True
if not win:
    print "Ισοπαλία"
show()
```

ΘΕΜΑ Γ_0_2010 (Αποτελέσματα αγώνα άλματος εις μήκος)

Σε κάποιο σχολικό αγώνα, για το άθλημα «Άλμα εις μήκος» καταγράφεται για κάθε αθλητή η καλύτερη έγκυρη επίδοσή του. Τιμής ένεκεν, πρώτος αγωνίζεται ο περσινός πρωταθλητής. Η Επιτροπή του αγώνα διαχειρίζεται τα στοιχεία των αθλητών που αγωνίστηκαν.

Να γράψετε αλγόριθμο ο οποίος:

Γ1. Να ζητάει το ρεκόρ αγώνων και να το δέχεται, εφόσον είναι θετικό και μικρότερο των 10 μέτρων.

Γ2. Να ζητάει τον συνολικό αριθμό των αγωνιζομένων και για κάθε αθλητή το όνομα και την επίδοσή του σε μέτρα με τη σειρά που αγωνίστηκε.

Γ3. Να εμφανίζει το όνομα του αθλητή με τη χειρότερη επίδοση.

Γ4. Να εμφανίζει τα ονόματα των αθλητών που κατέρριψαν το ρεκόρ αγώνων. Αν δεν υπάρχουν τέτοιοι αθλητές, να εμφανίζει το πλήθος των αθλητών που πλησίασαν το ρεκόρ αγώνων σε απόσταση όχι μεγαλύτερη των 50 εκατοστών.

Γ5. Να βρίσκει και να εμφανίζει τη θέση που κατέλαβε στην τελική κατάταξη ο περσινός πρωταθλητής.

Σημείωση: Να θεωρήσετε ότι κάθε αθλητής έχει έγκυρη επίδοση και ότι όλες οι επιδόσεις των αθλητών που καταγράφονται είναι διαφορετικές μεταξύ τους.

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:        2010_0_C
# Περιγραφή:    Αποτελέσματα αγώνα άλματος εις μήκος
# ΕνδΛύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ------------------------------------------------------------------

# Γ1)
record = input("Δώσε το ρεκόρ αγώνων = ")
while record >= 10 or record < 0 :
    print "Λάθος δεδομένα 0..10μ ..."
    record = input("Δώσε το ρεκόρ αγώνων = ")

# Γ2)
athlets = input("Δώσε το πλήθος των αθλητών = ")
old_name = raw_input("Δώσε το όνομα του περσινού πρωταθλητή = ")
old_jump = input("Δώσε το περσινό ρεκόρ = ")
count = 0
found = False

if old_jump > record:
    print "Πάνω απο το περσινό ρεκόρ"
    found = True
elif record - old_jump <= 0.5:
    count += 1

min_jump = 10
min_name = old_name

position_old_champ = 1
```

```
for athlete in range(athlets-1):
    name = raw_input("Δώσε το όνομα του αθλητή = ")
    jump = input("Δώσε το άλμα του αθλητή σε μέτρα = ")

    if jump < min_jump:
        min_name = name
        min_jump = jump

    if jump > record:
        print "Πάνω από το περσινό ρεκόρ πήδηξαν : ",
        print name, jump
        found = True
    elif record - jump <= 0.5:
        count += 1

    if jump > old_jump:
        position_old_champ += 1

print "Ο περσινός πρωταθλητής ήρθε στη θέση ", position_old_champ

# Γ3)
print "Χειρότερος αθλητής είναι ", min_name, min_jump
if not found:
    print "Κοντά στο ρεκόρ πήδηξαν (0.5meters) = ", count, ' αθλητές'
```

ΘΕΜΑ Δ_0_2010 (Αποτελέσματα αγώνα ιστιοπλοΐας)

Το ράλλυ Βορείων Σποράδων είναι ένας αγώνας ιστιοπλοΐας ανοικτής θάλασσας που γίνεται κάθε χρόνο. Στην τελευταία διοργάνωση συμμετείχαν 35 σκάφη που διαγωνίστηκαν σε διαδρομή συνολικής απόστασης 70 μιλίων. Κάθε σκάφος ανήκει σε μια από τις κατηγορίες C1, C2, C3. Επειδή στον αγώνα συμμετέχουν σκάφη διαφορετικών δυνατοτήτων, η κατάταξη δεν προκύπτει από τον «πραγματικό» χρόνο τερματισμού αλλά από ένα «σχετικό» χρόνο, που υπολογίζεται διαιρώντας τον «πραγματικό» χρόνο του σκάφους με τον «ιδανικό». Ο ιδανικός χρόνος είναι διαφορετικός για κάθε σκάφος και προκύπτει πολλαπλασιάζοντας την απόσταση της διαδρομής με τον δείκτη GPH του σκάφους. Ο δείκτης GPH αντιπροσωπεύει τον ιδανικό χρόνο που χρειάζεται το σκάφος για να καλύψει απόσταση ενός μιλίου.

Να κατασκευάσετε αλγόριθμο ο οποίος

Δ1. Να ζητάει για κάθε σκάφος:

- το όνομά του

- την κατηγορία του ελέγχοντας την ορθή καταχώρηση

- τον χρόνο (σε δευτερόλεπτα) που χρειάστηκε για να τερματίσει

- τον δείκτη GPH (σε δευτερόλεπτα).

Δ2. Να υπολογίζει τον σχετικό χρόνο κάθε σκάφους.

Δ3. Να εμφανίζει την κατηγορία στην οποία ανήκουν τα περισσότερα σκάφη.

Δ4. Να εμφανίζει για κάθε κατηγορία καθώς και για την γενική κατάταξη τα ονόματα των σκαφών που κερδίζουν μετάλλιο. (Μετάλλια απονέμονται στους 3 πρώτους κάθε κατηγορίας και στους 3 πρώτους της γενικής κατάταξης).

Σημείωση: Να θεωρήσετε ότι κάθε κατηγορία έχει διαφορετικό αριθμό σκαφών και τουλάχιστον τρία σκάφη.

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:        2010_0_D
# Περιγραφή:    Αποτελέσματα αγώνα ιστιοπλοΐας
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ------------------------------------------------------------------

# Δ1)
names = []
gph = []
real_times = []
cats = []
relative_times = []
k1 = k2 = k3 = 0
names1 = names2 = names3 = []
for i in range(35):
    names.append(raw_input("Δώσε το όνομα του σκάφους = "))
    cat = raw_input("Δώσε την κατηγορία του σκάφους = ")
    while cat.upper() not in ["C1", "C2", "C3"]:
        print "Λάθος δεδομένα .."
        cat = raw_input("Δώσε την κατηγορία του σκάφους = ")
```

```
    cats.append(cat)
    real_times.append(input("Δώσε τον πραγματικό χρόνο του σκάφους σε δευτ. = "))
    gph.append(input("Δώσε το δείκτη GPH του σκάφους σε δευτερόλεπτα = "))

    # Δ2)
    relative_times.append(real_times[i] / (70.0 * gph[i]))

    if cat == "C1":
        k1 += 1
    elif cat == "C2":
        k2 += 1
    else:
        k3 += 1

# Δ3)
print "Η πιο δημοφιλής κατηγορία είναι η : ",
if k1 > k2 and k1 > k3:
    print "C1"
elif k2 > k1 and k2 > k3:
    print "C2"
else:
    print "C3"

# Δ4)
print "Γενικά Αποτελέσματα"
for i in range(34):
    for j in range(34,i,-1):
        if relative_times[j] < relative_times[j-1]:
            relative_times[j], relative_times[j-1] = relative_times[j-1],
relative_times[j]
            names[j], names[j-1] = names[j-1], names[j]
            cats[j], cats[j-1] = cats[j-1], cats[j]
print names[0], names[1], names[2]
i = 0
count1 = count2 = count3 = 0
names1 = []
names2 = []
names3 = []
while count1 != 3 and count2 != 3 and count3 != 3:
    if cats[i] == "C1":
        count1 += 1
        names1.append(names[i])
    elif cats[i] == "C2":
        count2 += 1
        names2.append(names[i])
    elif cats[i] == "C3":
        count3 += 1
        names3.append(names[i])
    i += 1

print "Αποτελέσματα κατηγορίας C1"
print names1[0], names1[1], names1[2]
print "Αποτελέσματα κατηγορίας C2"
print names2[0], names2[1], names2[2]
print "Αποτελέσματα κατηγορίας C3"
print names3[0], names3[1], names3[2]
```

ΘΕΜΑ Γ_1_2010 (Έλεγχος Κωδικών πρόσβασης χρηστών)

Ένα σύστημα υπολογιστή χρησιμοποιεί για τον έλεγχο πρόσβασης των χρηστών του έναν πίνακα 1000 γραμμών και 3 στηλών με τα στοιχεία τους. Σε κάθε γραμμή του αποθηκεύει, στην πρώτη στήλη το όνομα πρόσβασης του χρήστη, στη δεύτερη στήλη το συνθηματικό του και στην τρίτη έναν από τους χαρακτήρες «Σ» ή «Α». (Ο χαρακτήρας «Σ» δηλώνει ότι το συνθηματικό συνεχίζει να ισχύει, ενώ ο χαρακτήρας «Α» δηλώνει ότι το συνθηματικό πρέπει να αλλάξει).

Θεωρήστε ότι υπάρχει ένα κύριο πρόγραμμα που υλοποιεί τα παραπάνω και καλεί τη διαδικασία ΕΛΕΓΧΟΣ η οποία ελέγχει την πρόσβαση του χρήστη στο σύστημα.

Να γράψετε τη διαδικασία ΕΛΕΓΧΟΣ η οποία να περιλαμβάνει:

Γ1. Τμήμα δηλώσεων.

Μονάδες 2

Κύριο τμήμα το οποίο:

Γ2. Διαβάζει το όνομα και το συνθηματικό του χρήστη. Ελέγχει αν το όνομα πρόσβασης και το συνθηματικό είναι έγκυρα, δηλαδή υπάρχουν στον πίνακα χρηστών και αναφέρονται στον ίδιο χρήστη. Αν υπάρχουν, εμφανίζει το μήνυμα «ΚΑΛΩΣ ΗΡΘΑΤΕ», διαφορετικά εμφανίζει το μήνυμα «ΛΑΘΟΣ ΟΝΟΜΑ ΠΡΟΣΒΑΣΗΣ Ή ΣΥΝΘΗΜΑΤΙΚΟ» και ζητά εκ νέου την εισαγωγή των δύο αυτών στοιχείων (ονόματος πρόσβασης και συνθηματικού) μέχρι να δοθούν έγκυρα στοιχεία.

Μονάδες 8

Γ3. Μετά την εμφάνιση του μηνύματος «ΚΑΛΩΣ ΗΡΘΑΤΕ» ελέγχει αν το συνθηματικό χρειάζεται αλλαγή. Αν χρειάζεται, ζητά από τον χρήστη την εισαγωγή νέου συνθηματικού δύο φορές (η δεύτερη ως επιβεβαίωση) μέχρις ότου το συνθηματικό και η επιβεβαίωσή του ταυτιστούν. Όταν ταυτιστούν, η διαδικασία αντικαθιστά το παλιό συνθηματικό με το νέο και τον αντίστοιχο χαρακτήρα «Α» της τρίτης στήλης με το «Σ».

Μονάδες 10

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:        2010_1_C
# Περιγραφή:    Έλεγχος Κωδικών πρόσβασης χρηστών
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ------------------------------------------------------------------

# Δοκιμαστικά δεδομένα εισόδου χρηστών
users = [
    ['alpha', 'a', 'Y'],
    ['bita', 'b', 'N'],
    ['gama', 'c', 'Y'],
    ['delta', 'd', 'N'],
    ['epsilon','e','Y'],
    ]


def check_password():
    found = False
    while not found:
        user = raw_input("Δώσε το όνομα χρήστη = ")
        password = raw_input("Δώσε τον κωδικό χρήστη = ")
        i = 0
        while i <= len(users)-1 and found == False:
            if users[i][0] == user:
                if users[i][1] == password:
                    found = True
                else:
                    found = False
                    i += 1
            else:
                found = False
                i += 1
        if found:
            print "Καλώς ήρθατε πάλι"
            if users[i][2].upper() == 'N':
                print "Ο παλαιός κωδικός σας έληξε....."
                new_password1 = raw_input("Εισάγετε νέο κωδικό = ")
                new_password2 = raw_input("Εισάγετε νέο κωδικό = ")
                while new_password1 != new_password2:
                    print "Οι κωδικοί δεν ταιριάζουν ξαναπροσπαθήστε"
                    new_password1 = raw_input("Εισάγετε νέο κωδικό = ")
                    new_password2 = raw_input("Εισάγετε νέο κωδικό = ")
                print "Επιτυχής εισαγωγή νέου κωδικού"
                users[i][1] = new_password1
                users[1][2] = 'Y'
        else:
            print "Λάθος κωδικός ή όνομα χρήστη"

check_password()
```

ΘΕΜΑ Δ_1_2010 (Μοντέλο προσομοίωσης μολυσματικής ασθένειας)

Ερευνητές που ασχολούνται με μοντέλα προσομοίωσης εξάπλωσης επιδημιών χρησιμοποιούν για τις μελέτες τους ένα αριθμητικό πίνακα Μ[5000]. Κάθε κελί του πίνακα αυτού αντιπροσωπεύει ένα άτομο σε μια περιοχή 5.000 κατοίκων στην οποία υπάρχουν εστίες μιας συγκεκριμένης μολυσματικής ασθένειας (επιδημίας). Από σύμβαση η τιμή μηδέν 0 σε ένα κελί αντιπροσωπεύει ένα υγιές άτομο, ενώ η τιμή -1 αντιπροσωπεύει ένα άτομο που έχει τη συγκεκριμένη ασθένεια (μολυσμένο άτομο). Κάθε άτομο έρχεται σε επαφή με τα γειτονικά του και η ασθένεια μπορεί να μεταδοθεί από τον ένα στον άλλο. (Γειτονικά χαρακτηρίζονται δύο άτομα, όταν τα κελιά του πίνακα που τα αντιπροσωπεύουν έχουν μια κοινή πλευρά).

Θεωρήστε ότι δίνεται ο πίνακας Μ που περιέχει ήδη έναν αριθμό μολυσμένων ατόμων. Να υλοποιήσετε αλγόριθμο ο οποίος:

Δ1. Υπολογίζει και εμφανίζει με κατάλληλο μήνυμα τον συνολικό αριθμό των μολυσμένων ατόμων που υπάρχουν στο σύνολο του πληθυσμού.

Μονάδες 4

Δ2. Αποθηκεύει σε κάθε κελί του πίνακα Μ που αντιπροσωπεύει ένα υγιές άτομο έναν αριθμό ο οποίος δείχνει με πόσα μολυσμένα άτομα γειτονεύει το υγιές.

Μονάδες 8

Δ3. Βρίσκει αν υπάρχει έστω και μία «σημαντική» εστία μόλυνσης. Αν υπάρχει, εμφανίζει το μήνυμα «Υπάρχει σημαντική εστία μόλυνσης» μαζί με τη θέση του πρώτου κελιού της εστίας. Αν δεν υπάρχει, εμφανίζει το μήνυμα «Δεν υπάρχει σημαντική εστία μόλυνσης». (Μια εστία μόλυνσης χαρακτηρίζεται σημαντική, όταν δύο ή περισσότερα μολυσμένα άτομα βρίσκονται σε συνεχόμενα γειτονικά κελιά).

Μονάδες 8

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:       2010_1_D
# Περιγραφή:   Μοντέλο προσομοίωσης μολυσματικής ασθένειας
# Ενδλύση:     TasosChatzipapadopoulos
# Python vs:   2.7.10
# Copyright:   (c) 2018
# Url:         http://users.sch.gr/chatzipap
# ------------------------------------------------------------------

# Δεδομενα Δοκιμής
m = [0, -1, 0, -1, 0, 0, -1, -1, 0, -1, 0, 0, -1, 0]


# Δ1)
print m
# Ερώτημα Δ1
count = 0
for i in range(len(m)):
    if m[i] == -1:
        count += 1

print "Μολυσμένα Άτομα = ", count

# Ερώτημα Δ2)
```

```
count = 0
before = after = 0
for i in range(1, len(m)-1):
    before = i -1
    after = i + 1
    if m[i] == 0:
        if m[before] == -1:
            count += 1
        if m[after] == -1:
            count += 1
        m[i] = count
    count = 0
# Το 1ο στοιχείο έχει μόνο επόμενο
if m[0] == 0:
    if m[1] == -1:
        m[0] = 1
# Το τελευταίο στοιχείο έχει μόνο προηγούμενο
if m[len(m)-1] == 0:
    if m[len(m)-2] == -1:
        m[len(m)-1] = 1
print "Χάρτης μολυνσης = ", m

# Ερώτημα Δ3)
count = 0
found = False
i = 0
while not found and i <= len(m)-1:
    if m[i] == -1:
        count += 1
    else:
        count = 0
    if count == 2:
        found = True
        position = i - 1
    i += 1
if found:
    print "Εστία μόλυνσης βρέθηκε στη θέση = ", position
else:
    print "Δε βρέθηκε εστία μόλυνσης"
```

ΘΕΜΑ Γ_0_2011 (Επιτυχόντες εξετάσεων ΑΣΕΠ)

Στις εξετάσεις του ΑΣΕΠ οι υποψήφιοι εξετάζονται σε τρεις θεματικές ενότητες. Ο βαθμός κάθε θεματικής ενότητας είναι από 1 έως 100. Η συνολική βαθμολογία κάθε υποψηφίου προκύπτει από τον μέσο όρο των βαθμών του στις τρεις θεματικές ενότητες. Ο υποψήφιος θεωρείται ως επιτυχών, αν η συνολική βαθμολογία του είναι τουλάχιστον 55 και ο βαθμός του σε κάθε θεματική ενότητα είναι τουλάχιστον 50.

Να γράψετε αλγόριθμο ο οποίος:

Για κάθε υποψήφιο:

Γ1. Να διαβάζει το όνομά του και τους βαθμούς του σε καθεμία από τις τρεις θεματικές ενότητες. (Δεν απαιτείται έλεγχος εγκυρότητας δεδομένων).

Μονάδες 2

Γ2. Να εμφανίζει τον μεγαλύτερο από τους βαθμούς που πήρε στις τρεις θεματικές ενότητες.

Μονάδες 5

Γ3. Να εμφανίζει το όνομα και τη συνολική βαθμολογία του στην περίπτωση που είναι επιτυχών.

Μονάδες 4

Γ4. Ο αλγόριθμος να τερματίζει όταν δοθεί ως όνομα η λέξη “ΤΕΛΟΣ”.

Μονάδες 4

Γ5. Στο τέλος να εμφανίζει το όνομα του επιτυχόντα με τη μικρότερη συνολική βαθμολογία. Θεωρήστε ότι είναι μοναδικός.

Μονάδες 5

```
# ----------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ----------------------------------------------------------------------
# Όνομα:        2011_0_C
# Περιγραφή:    Επιτυχόντες εξετάσεων ΑΣΕΠ
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ----------------------------------------------------------------------

flag = ""
xmin = 100
name_min = ""
name = raw_input("Δώσε το όνομα του υποψηφίου (END) για τέλος = ")
while name.upper() != 'END':
    # Ερώτημα Γ1
    a, b, c = input("Δώσε 3 βαθμολογίες a,,b,c =")

    # Ερώτημα Γ2
    xmax = a
    if b > xmax:
        xmax = b
    if c > xmax:
        xmax = c
    print "Μεγαλύτερη βαθμολογία = ", xmax

    # Ερώτημα Γ3
    mo = (a + b + c) / 3.0
    if mo >= 55:
        if a >= 50 and b >= 50 and c >= 50:
            print "Επιτυχών ", name, mo

    # Ερώτημα Γ4
    if mo < xmin:
        xmin = mo
        name_min = name

    name = raw_input("Δώσε το όνομα του υποψηφίου (END) για τέλος = ")

print "Μικρότερη Βαθμολογία = ", xmin, " από ", name_min
```

ΘΕΜΑ Δ_0_2011 (Εκλογή αρχηγού ποδοσφαιρικής ομάδας)

Στην αρχή της ποδοσφαιρικής περιόδου οι 22 παίκτες μιας ομάδας, οι οποίοι αριθμούνται από 1 έως 22, ψηφίζουν για τους 3 αρχηγούς που θα τους εκπροσωπούν. Κάθε παίκτης μπορεί να ψηφίσει όσους συμπαίκτες του θέλει, ακόμα και τον εαυτό του. Τα αποτελέσματα της ψηφοφορίας καταχωρίζονται σε έναν πίνακα ΨΗΦΟΣ με 22 γραμμές και 22 στήλες, έτσι ώστε το στοιχείο ΨΗΦΟΣ[i,j] να έχει την τιμή 1, όταν ο παίκτης με αριθμό i έχει ψηφίσει τον παίκτη με αριθμό j, και τιμή 0 στην αντίθετη περίπτωση.

Να γράψετε αλγόριθμο ο οποίος:

Δ1. Να διαβάζει τα στοιχεία του πίνακα ΨΗΦΟΣ και να ελέγχει την ορθότητά τους με αποδεκτές τιμές 0 ή 1.

Μονάδες 4

Δ2. Να εμφανίζει το πλήθος των παικτών που δεν ψήφισαν κανέναν.

Μονάδες 4

Δ3. Να εμφανίζει το πλήθος των παικτών που ψήφισαν τον εαυτό τους.

Μονάδες 4

Δ4. Να βρίσκει τους 3 παίκτες που έλαβαν τις περισσότερες ψήφους και να εμφανίζει τους αριθμούς τους και τις ψήφους που έλαβαν. Θεωρήστε ότι δεν υπάρχουν ισοψηφίες.

Μονάδες 8

```
# ----------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ----------------------------------------------------------------------
# Όνομα:        2011_0_D
# Περιγραφή:    Εκλογή αρχηγού ποδοσφαιρικής ομάδας
# ΕνδΛύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ----------------------------------------------------------------------

# Ερώτημα Δ1
votes = []
for i in range(4):
    votes.append([])
    for j in range(4):
        vote = input("Κε" + str(i) + " θέλεις να ψηφίσεις για παίκτη (0,1)= "\
                     + str(j) + "?")
        while vote not in [0, 1]:
            print "Λάθος δεδομένα ξαναπροσπάθησε ....."
            vote = input("Κε" + str(i) + " θέλεις να ψηφίσεις για παίκτη (0,1)= "\
                         + str(j) + "?")
        votes[i].append(vote)

# Ερώτημα Δ2
count = 0
for i in range(4):
    vote = 0
    for j in range(4):
        if votes[i][j] == 1:
            vote += 1
```

```
    if vote == 0 :
        count += 1
print "Πλήθος παικτών που δεν ψήφισαν κανέναν ", count, " παίκτες"

# Ερώτημα Δ3
count = 0
for i in range(4):
    if votes[i][i] == 1:
        count += 1
print "Πλήθος παικτών που ψήφισαν τον εαυτό τους ", count, " παίκτες"

# Ερώτημα Δ4
count = [0] * 4
for j in range(4):
    for i in range(4):
        if votes[i][j] == 1:
            count[j] += 1

num = [i for i in range(4)]  # Εκτός ύλης πίνακας αριθμών φανέλλας
# Παράλληλη Ταξινόμηση
for i in range(3):
    for j in range(3, i, -1):
        if count[j-1] < count[j]:
            count[j-1], count[j] = count[j], count[j-1]
            num[j-1], num[j] = num[j], num[j-1]

for i in range(3):
    print num[i], count[i]
```

ΘΕΜΑ Γ_1_2011 (Ανεφοδιασμός πρατηρίου υγρών καυσίμων)

Ένα πρατήριο υγρών καυσίμων διαθέτει έναν τύπο καυσίμου που αποθηκεύεται σε δεξαμενή χωρητικότητας 10.000 λίτρων. Να αναπτύξετε αλγόριθμο ο οποίος:

Γ1. να διαβάζει την ποσότητα (σε λίτρα) του καυσίμου που υπάρχει αρχικά στη δεξαμενή μέχρι να δοθεί έγκυρη τιμή.

Μονάδες 2

Για κάθε όχημα που προσέρχεται στο πρατήριο:

Γ2. να διαβάζει τον τύπο του οχήματος (“Β” για βυτιοφόρο όχημα που προμηθεύει το πρατήριο με καύσιμο και “Ε” για επιβατηγό όχημα που προμηθεύεται καύσιμο από το πρατήριο).

Μονάδες 2

Γ3. Αν το όχημα είναι βυτιοφόρο τότε να γεμίζει τη δεξαμενή μέχρι την πλήρωσή της. (μονάδες 3) Αν το όχημα είναι επιβατηγό τότε να διαβάζει την ποσότητα καυσίμου την οποία θέλει να προμηθευτεί (μονάδες 2) και, αν υπάρχει επάρκεια καυσίμου στη δεξαμενή, τότε το επιβατηγό όχημα να εφοδιάζεται με τη ζητούμενη ποσότητα καυσίμου, διαφορετικά το όχημα να μην εξυπηρετείται (μονάδες 3).

Μονάδες 8

Γ4. Η επαναληπτική διαδικασία να τερματίζεται, όταν αδειάσει η δεξαμενή του πρατηρίου ή όταν δεν εξυπηρετηθούν τρία διαδοχικά επιβατηγά οχήματα.

Μονάδες 4

Γ5. Στο τέλος ο αλγόριθμος να εμφανίζει:

α. τη μέση ποσότητα καυσίμου ανά επιβατηγό όχημα που εξυπηρετήθηκε

β. τη συνολική ποσότητα καυσίμου με την οποία τα βυτιοφόρα ανεφοδίασαν τη δεξαμενή.

Μονάδες 4

Σημειώσεις: • Δεν απαιτείται έλεγχος εγκυρότητας για τον τύπο του οχήματος. • Θεωρήστε ότι στο πρατήριο προσέρχεται ένα τουλάχιστον επιβατηγό όχημα για το οποίο η ποσότητα καυσίμου στη δεξαμενή επαρκεί.

```
# ----------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ----------------------------------------------------------------------
# Όνομα:        2011_1_C
# Περιγραφή:    Ανεφοδιασμός πρατηρίου υγρών καυσίμων
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ----------------------------------------------------------------------

# Ερώτημα Γ1
tank = input("Δώσε αρχική ποσότητα = ")
while tank <= 0 or tank >= 10000:
    start = input("Δώσε αρχική ποσότητα = ")

# Ερώτημα Γ2
not_served = 0
served = 0
fuel = 0
total_in = 0
while not_served != 3 or tank == 0:
    vehicle = raw_input("Δώσε τον τύπο του οχήματος (T)Tanker, (C)Car = ")
    vehicle = vehicle.upper()
    # Ερώτημα Γ3
    if vehicle == "T":
        total_in += 10000 - tank
        tank = 10000
    if vehicle == "C":
        quantity = input("Δώσε ποσότητα καυσίμου για ανεφοδιασμό = ")
        if tank >= quantity:
            tank -= quantity
            served += 1
            fuel += quantity
        else:
            # Ερώτημα Γ4
            print "Απαίτηση οχήματος ανέφικτη"
            not_served += 1
    print "Στάθμη δεξαμενής καυσίμου = ", tank

# Ερώτημα Γ5
average = fuel / served * 1.0
print "Μέσος όρος καυσίμου ανά όχημα = ", average
print "Συνολική ποσότητα εισαγωγής καυσίμου = ", total_in
```

ΘΕΜΑ Δ_1_2011 (Κέρδη 5ετίας εταιριών)

Ένας όμιλος αποτελείται από 20 εταιρίες. Να γράψετε πρόγραμμα το οποίο:

Δ1. να περιλαμβάνει τμήμα δηλώσεων.

Μονάδες 2

Δ2. να διαβάζει τα ονόματα των εταιριών του ομίλου και τα κέρδη τους για κάθε ένα από τα έτη 2001 έως και 2005. (Θεωρήστε ότι τα κέρδη είναι θετικοί αριθμοί.)

Μονάδες 2

Δ3. για κάθε εταιρία του ομίλου να καλεί συνάρτηση για τον υπολογισμό του συνολικού κέρδους της εταιρίας στην πενταετία. Στη συνέχεια να υπολογίζει και να εμφανίζει το μέσο ετήσιο κέρδος του ομίλου.

Μονάδες 5

Δ4. για κάθε εταιρία να βρίσκει την τριετία με το μεγαλύτερο συνολικό κέρδος και να εμφανίζει το όνομα της εταιρίας και το πρώτο έτος της συγκεκριμένης τριετίας. (Θεωρήστε ότι η τριετία αυτή είναι μοναδική.)

Μονάδες 5

Δ5. Να κατασκευάσετε τη συνάρτηση που θα χρησιμοποιήσετε στο ερώτημα Δ3.

Μονάδες 6

```
# ----------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ----------------------------------------------------------------------
# Όνομα:       2011_1_D
# Περιγραφή:   Κέρδη 5ετίας εταιριών
# Ενδλύση:     TasosChatzipapadopoulos
# Python vs:   2.7.10
# Copyright:   (c) 2018
# Url:         http://users.sch.gr/chatzipap
# ----------------------------------------------------------------------

# Ερώτημα 1 & 2
names = []
profits = []
for i in range(20):
    names.append(raw_input("Δώσε το όνομα της εταιρείας = "))
    profits.append([])
    year = 2001
    for j in range(5):
        profits[i].append(input("Δώσε τα ετήσια κέρδη της εταιρείας " + str(year) + "
= "))
        year += 1

# Ερώτημα 5
def calc_profit(name):
    i = 0
    position = -1
    while position == -1 and i <= 2:
        if names[i] == name:
            position = i
        else:
            i += 1
    if position != -1:
```

```
        total_profit = 0
        for j in range(5):
            total_profit += profits[position][j]
        return total_profit
    else:
        print "Όνομα εταιρείας = ", name, " δεν βρέθηκε"

# Ερώτημα 3
total = 0
for name in names:
    name_profit = calc_profit(name)
    print name, name_profit
    total += name_profit
print "Συνολικά κέρδη όλων των εταιρείών = ", total

# Ερώτημα 4
for i in range(20):
    max3 = 0  # Σύνολο κερδών τριετίας
    max_year = 0  # Καλύτερος 1ος χρόνος τριετίας
    start = 0  # Θα αυξάνει κάθε αλλαγή τριετίας κατά ένα
    for year in range(3):  # Σύνολο 3 τριετίες
        xsum = 0
        for j in range(start, start + 3):  # Κάθε τριετία αλλάζει όρια
            xsum += profits[i][j]
        if xsum > max3:
            max_year = year
        max3 = xsum
        start += 1
    # Εμφάνιση εταιρείας και καλυτερης τριετίας
    print names[i], max3, max_year + 2001
```

ΘΕΜΑ Γ_0_2012 (Επιδοτήσεις Δημοσίων έργων)

Δημόσιος οργανισμός διαθέτει ένα συγκεκριμένο ποσό για την επιδότηση επενδυτικών έργων. Η επιδότηση γίνεται κατόπιν αξιολόγησης και αφορά δύο συγκεκριμένες κατηγορίες έργων με βάση τον προϋπολογισμό τους. Οι κατηγορίες και τα αντίστοιχα ποσοστά επιδότησης επί του προϋπολογισμού φαίνονται στον παρακάτω πίνακα.

Κατηγορία έργου	Προϋπολογισμός έργου σε ευρώ	Ποσοστό Επιδότησης
Μικρή	200.000 – 299.999	60%
Μεγάλη	300.000 – 399.999	70%

Η εκταμίευση των επιδοτήσεων των αξιολογηθέντων έργων γίνεται με βάση τη χρονική σειρά υποβολής τους. Μετά από κάθε εκταμίευση μειώνεται το ποσό που διαθέτει ο οργανισμός.

Να αναπτύξετε αλγόριθμο ο οποίος:

Γ1. Να διαβάζει το ποσό που διαθέτει ο οργανισμός για το πρόγραμμα επενδύσεων συνολικά, ελέγχοντας ότι το ποσό είναι μεγαλύτερο από 5.000.000 ευρώ.

Μονάδες 2

Γ2. Να διαβάζει το όνομα κάθε έργου. Η σειρά ανάγνωσης είναι η σειρά υποβολής των έργων. Η επαναληπτική διαδικασία να τερματίζεται, όταν αντί για όνομα έργου δοθεί η λέξη «ΤΕΛΟΣ», ή όταν το διαθέσιμο ποσό έχει μειωθεί τόσο, ώστε να μην είναι δυνατή η επιδότηση ούτε ενός έργου μικρής κατηγορίας. Για κάθε έργο, αφού διαβάσει το όνομά του, να διαβάζει και τον προϋπολογισμό του (δεν απαιτείται έλεγχος εγκυρότητας του προϋπολογισμού).

Μονάδες 6

Γ3. Για κάθε έργο να ελέγχει αν το διαθέσιμο ποσό καλύπτει την επιδότηση, και μόνον τότε να γίνεται η εκταμίευση του ποσού. Στη συνέχεια, να εμφανίζει το όνομα του έργου και το ποσό της επιδότησης που δόθηκε.

Μονάδες 6

Γ4. Να εμφανίζει το πλήθος των έργων που επιδοτήθηκαν από κάθε κατηγορία καθώς και τη συνολική επιδότηση που δόθηκε σε κάθε κατηγορία.

Μονάδες 4

Γ5. Μετά το τέλος της επαναληπτικής διαδικασίας να εμφανίζει το ποσό που δεν έχει διατεθεί, μόνο αν είναι μεγαλύτερο του μηδενός.

Μονάδες 2

```
# ------------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------------
# Όνομα:        2012_0_C
# Περιγραφή:    Επιδοτήσεις Δημοσίων έργων
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ------------------------------------------------------------------------

# Ερώτημα 1
amount = input("Δώστε το ποσό της επένδυσης = ")
while amount <= 5000000:
    print "Είναι μικρό το ποσό ξαναπροσπαθήστε ... "
    amount = input("Δώστε το ποσό της επένδυσης = ")

# Ερώτημα 2
small = 0
large = 0
small_sum = 0
large_sum = 0

project = raw_input("Δώστε το όνομα του έργου = ")
while project.upper() != "END" and amount > 200000 * 0.6:
    budget = input("Δώστε τον προϋπολογισμό του έργου = ")
    if budget >= 200000 and budget <= 299999:
        support = budget * 0.6
        flag = 1
    elif budget >= 300000 and budget <= 399999:
        support = budget * 0.7
        flag = 2
    else:
        support = 0
    # Ερώτημα 3
    if support != 0:
        if amount >= support:
            amount -= support
            if flag == 1:
                small += 1
                small_sum += support
            elif flag == 2:
                large += 1
                large_sum += support
            print "Το έργο ", project, " λαμβάνει ", support, " υποστήριξη"
    project = raw_input("Δώστε το όνομα του έργου = ")

# Ερώτημα 4
print "Μικρής Κλίμακας ", small, " Συνολική Στήριξη ", small_sum
print "Μεγάλης Κλίμακας ", large, " Συνολική Στήριξη ", large_sum

# Ερώτημα 5
if amount > 0:
    print "Χρήματα που περίσσεψαν για επενδύσεις ", amount
```

ΘΕΜΑ Δ_0_2012 (Κατανάλωση Ενέργειας πόλεων-πελατών-έτους)

Μια εταιρεία ασχολείται με εγκαταστάσεις φωτοβολταϊκών συστημάτων, με τα οποία οι πελάτες της έχουν τη δυνατότητα αφενός να παράγουν ηλεκτρική ενέργεια για να καλύπτουν τις ανάγκες της οικίας τους, αφετέρου να πωλούν την πλεονάζουσα ενέργεια προς 0,55€/kWh, εξασφαλίζοντας επιπλέον έσοδα. Η εταιρεία αποφάσισε να ερευνήσει τις εγκαταστάσεις που πραγματοποίησε την προηγούμενη χρονιά σε δέκα (10) πελάτες που βρίσκονται ο καθένας σε διαφορετική πόλη της Ελλάδας.

Να αναπτύξετε πρόγραμμα σε ΓΛΩΣΣΑ το οποίο:

Δ1. α. Να περιλαμβάνει κατάλληλο τμήμα δηλώσεων.

(μονάδα 1)

β. Να διαβάζει για κάθε πελάτη το όνομά του και το όνομα της πόλης στην οποία διαμένει και να τα αποθηκεύει στον δισδιάστατο πίνακα ΟΝ[10,2].

(μονάδα 1)

γ. Να διαβάζει το ποσό της ηλεκτρικής ενέργειας σε kWh που παρήγαγαν τα φωτοβολταϊκά συστήματα κάθε πελάτη, καθώς και το ποσό της ηλεκτρικής ενέργειας που κατανάλωσε κάθε πελάτης για κάθε μήνα του έτους, και να τα αποθηκεύει στους πίνακες Π[10,12] για την παραγωγή και Κ[10,12] για την κατανάλωση αντίστοιχα (δεν απαιτείται έλεγχος εγκυρότητας των δεδομένων).

(μονάδες 2) Μονάδες 4

Δ2. Να υπολογίζει την ετήσια παραγωγή και κατανάλωση ανά πελάτη καθώς και τα ετήσια έσοδά του σε ευρώ (€). Θεωρήστε ότι για κάθε πελάτη η ετήσια παραγόμενη ηλεκτρική ενέργεια είναι μεγαλύτερη ή ίση της ενέργειας που έχει καταναλώσει.

Μονάδες 4

Δ3. Να εμφανίζει το όνομα της πόλης στην οποία σημειώθηκε η μεγαλύτερη παραγωγή ηλεκτρικού ρεύματος.

Μονάδες 3

Δ4. Να καλεί κατάλληλο υποπρόγραμμα με τη βοήθεια του οποίου θα εμφανίζονται τα ετήσια έσοδα κάθε πελάτη κατά φθίνουσα σειρά. Να κατασκευάσετε το υποπρόγραμμα που χρειάζεται για το σκοπό αυτό.

Μονάδες 5

Δ5. Να εμφανίζει τον αριθμό του μήνα με τη μικρότερη παραγωγή ηλεκτρικής ενέργειας. Θεωρήστε ότι υπάρχει μόνο ένας τέτοιος μήνας.

Μονάδες 4

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:        2012_0_D
# Περιγραφή:    Κατανάλωση Ενέργειας πόλεων-πελατών-έτους
# ΕνδΛύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ------------------------------------------------------------------------------

# Ερώτημα Δ1
name = []
```

```
for i in range(10):
    name.append([])
    name[i].append(raw_input("Δώσε το όνομα κτου καταναλωτή = "))
    name[i].append(raw_input("Σε ποια πόλη κατοικεί = "))

kwh_out = []
kwh_in = []
for i in range(10):
    kwh_out.append([])
    kwh_in.append([])
    for j in range(12):
        kwh_out[i].append(input("Δώσε την παραγωγή ενέργειας = "))
        kwh_in[i].append(input("Δώσε την κατανάλωση ενέργειας = "))

# Ερώτημα Δ2
sum_out = sum_in = [0] * 10
income = []
for i in range(10):
    for j in range(12):
        sum_out[i] += kwh_out[i][j]
        sum_in[i] += kwh_in[i][j]
    income.append((sum_out[i] - sum_in[i]) * 0.55)

# Ερώτημα Δ3
print "Πόλη(εις) με τη μέγιστη κατανάλωση ενέργειας .."
max_out = sum_out[0]
for i in range(1,10):
    if sum_out[i] > max_out:
        max_out = sum_out[i]
for i in range(10):
    if max_out == sum_out[i]:
        print name[i][1]

# Ερώτημα Δ4
# Αντίγραφα των λιστών ονομάτων και εισοδήματος
# πριν την ταξινόμηση
names = [name[i][0] for i in range(10)] # list comprehension Εκτός ύλης
money = [income[i] for i in range(10)]  # list comprehension Εκτός ύλης
def sort_income():
    for i in range(9):
        for j in range(9,i,-1):
            if money[j] > money[j-1]:
                money[j], money[j-1] = money[j-1], money[j]
                names[j], names[j-1] = names[j-1], names[j]
    for i in range(10):
        print names[i], money[i]
sort_income()

# Ερώτημα Δ5
month_out = []
for j in range(12):
    month_out.append(0)
    for i in range(10):
        month_out[j] += kwh_out[i][j]

min_out = month_out[0]
```

```
jmin_out = 0
for j in range(1, 12):
    if month_out[j] < min_out:
        jmin_out = month_out[j]
        jmin_out = j
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
          'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']

print "Ο Μήνας με την μικρότερη παραγωγή ενέργειας είναι ", months[jmin_out]
```

ΘΕΜΑ Γ_1_2012 (Κρυπτογράφηση κειμένου)

Η κρυπτογράφηση χρησιμοποιείται για την προστασία των μεταδιδόμενων πληροφοριών. Ένας απλός αλγόριθμος κρυπτογράφησης χρησιμοποιεί την αντιστοίχηση κάθε γράμματος ενός κειμένου σε ένα άλλο γράμμα της αλφαβήτου. Για το σκοπό αυτό δίνεται πίνακας ΑΒ[2,24], ο οποίος στην πρώτη γραμμή του περιέχει σε αλφαβητική σειρά τους χαρακτήρες από το Α έως και το Ω. Στη δεύτερη γραμμή του βρίσκονται οι ίδιοι χαρακτήρες, αλλά με διαφορετική σειρά. Κάθε χαρακτήρας της πρώτης γραμμής κρυπτογραφείται στον αντίστοιχο χαρακτήρα της δεύτερης γραμμής, που βρίσκεται στην ίδια στήλη. Επίσης, δίνεται πίνακας ΚΕΙΜ[500], ο οποίος περιέχει αποθηκευμένο με κεφαλαία ελληνικά γράμματα το προς κρυπτογράφηση κείμενο. Κάθε χαρακτήρας του κειμένου βρίσκεται σε ένα κελί του πίνακα ΚΕΙΜ[500]. Οι λέξεις του κειμένου χωρίζονται με έναν χαρακτήρα κενό (΄ ΄), ενώ στο τέλος του κειμένου μπορεί να υπάρχουν χαρακτήρες κενό (΄ ΄), μέχρι να συμπληρωθεί ο πίνακας.

Να αναπτύξετε αλγόριθμο ο οποίος:

Γ1. Να εμφανίζει το πλήθος των χαρακτήρων κενό (΄ ΄), που υπάρχουν μετά το τέλος του κειμένου στον πίνακα ΚΕΙΜ[500]. Αν δεν υπάρχει χαρακτήρας κενό μετά τον τελευταίο χαρακτήρα του μη κρυπτογραφημένου κειμένου, τότε να εμφανίζεται το μήνυμα: «Το μήκος του κειμένου είναι 500 χαρακτήρες». Θεωρήστε ότι ο πίνακας ΚΕΙΜ[500] περιέχει τουλάχιστον μία λέξη.

Μονάδες 5

Γ2. Να κρυπτογραφεί τους χαρακτήρες του πίνακα ΚΕΙΜ[500] στον πίνακα ΚΡΥΠ[500], με βάση τον πίνακα ΑΒ[2,24]. Η κρυπτογράφηση να τερματίζεται με το τέλος του κειμένου. Δίνεται ότι κάθε χαρακτήρας κενό, που υπάρχει στον πίνακα ΚΕΙΜ[500], παραμένει χαρακτήρας κενό στον πίνακα ΚΡΥΠ[500].

Μονάδες 7

Γ3. Να εμφανίζει το πλήθος των λέξεων του κειμένου, καθώς και το πλήθος των χαρακτήρων που έχει η μεγαλύτερη λέξη του κειμένου στον πίνακα ΚΡΥΠ[500]. Θεωρήστε ότι η μεγαλύτερη λέξη είναι μοναδική.

Μονάδες 8

```
# ---------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ---------------------------------------------------------------------
# Όνομα:       2012_1_C
# Περιγραφή:   Κρυπτογράφηση κειμένου
# Ενδλύση:     TasosChatzipapadopoulos
# Python vs:   2.7.10
# Copyright:   (c) 2018
# Url:         http://users.sch.gr/chatzipap
# ---------------------------------------------------------------------------

# Ερώτημα Δ1
# Το θέμα αυτό ευνοεί να παίξουμε λίγο ξεφεύγοντας από
# τα στενά όρια της εκφώνησης που είναι προσαρμοσμένη στην ΑΕΠΠ
# Για να μην υπάρχουν προβλήματα με τους ελληνικούς χαρακτήρες
# θα χρησιμοποιήσουμε λατινικό αλφάβητο

alpha = ['A', 'B', 'C', 'D', 'E',
         'F', 'G', 'H', 'I', 'J',
         'K', 'L', 'M', 'N', 'O',
         'P', 'Q', 'R', 'S', 'T',
         'U', 'V', 'W', 'X', 'Y',
```

```
        'Z']


# Δημιουργία της κρτυπτογραφημένης λίστας (με τυχαίο τρόπο)
def create_code():
    bita = alpha[:]
    import random
    code_alpha = []
    n = len(bita)
    while n > 0:
        item = bita.pop(random.randint(0,n-1))
        code_alpha.append(item)
        n = len(bita)
    return code_alpha

code_alpha = create_code()

# Δημιουργία του πίνακα AB
ab = []
for i in range(26):
    ab.append([alpha[i], code_alpha[i]])
print ab

# Καθόσον η Python χειρίζεται μια χαρά τα string δίνουμε το προς κρυπτογράφηση
κείμενο ως τέτοιο
text = '''Python is high level programming language for general purpose programming
created by Guido van Rossum An interpreted language  Python has a design philosophy
that emphasizes code readability notably using whitespace indentation to delimit code
blocks rather than curly brackets or keywords and a syntax that allows programmers to
express concepts in fewer lines of code than other languages The language provides
constructs intended to enable writing clear programs on both a small and large scale
'''
print text, len(text)

# Ερώτημα Γ1
flag = False
n = len(text) - 1
i = n
blanks = 0
only_letters = 500
while not flag and i >= 0:
    if text[i] == ' ':
        blanks += 1
        i -= 1
    else:
        flag = True
if blanks == 0:
    print "Μέγεθος Κειμένου = 500"
else:
    only_letters = 500 - blanks
    print "Μέγεθος κειμένου = ", only_letters, " και ", blanks, " κενά στο τέλος"


# Ερώτημα Γ2
def find_letter(x):
    pos = -1
```

```
    for j in range(26):
        if x.upper() == ab[j][0]:
            pos = j
    if pos == -1:
        return False
    else:
        return pos
crypto = ''
for letter in text:
    found = find_letter(letter)
    if found is not False:
        crypto += ab[found][1]
    else:
        crypto += letter
print "Κρυπτογραφημένο κείμενο ...."
print crypto


# Ερώτημα Γ3
def count_words(text):
    words = 0
    for i in range(only_letters - 1):
        if text[i] == ' ':
            words += 1
    return words + 1  # Δύο κενά = τρεις λέξεις.


def split_in_words(text):
    words_list = []
    word = ''
    for i in range(only_letters - 1):
        if text[i] in [' ', '\n']:  # Μέριμνα για το χαρακτήρα αλλαγής γραμμής
            words_list.append(word)
            word = ''
        else:
            word += text[i]
    return words_list


def max_word(words):
    xmax = len(words[0])
    for word in words:
        if len(word) > xmax:
            xmax = len(word)
    return xmax

print "Πλήθος λέξεων = ", count_words(text)
print "Μήκος μεγαλύτερης λέξης = ", max_word(split_in_words(text))
```

ΘΕΜΑ Δ_1_2012 (Σφυγμομέτρηση τηλεθέασης εβδομάδας)

Εταιρεία που ασχολείται με μετρήσεις τηλεθέασης καταγράφει στοιχεία, ανά ημέρα και για χρονικό διάστημα μίας εβδομάδας, τα οποία αφορούν την τηλεθέαση των κεντρικών δελτίων ειδήσεων που προβάλλονται από πέντε (5) τηλεοπτικούς σταθμούς. Για τη διευκόλυνση της στατιστικής επεξεργασίας των δεδομένων να αναπτύξετε πρόγραμμα το οποίο:

Δ1. Να περιλαμβάνει τμήμα δηλώσεων.

Μονάδες 2

Δ2. Για κάθε έναν από τους τηλεοπτικούς σταθμούς να δέχεται το όνομά του και το πλήθος των τηλεθεατών που παρακολούθησαν το κεντρικό δελτίο ειδήσεων κάθε μέρα της εβδομάδας, από Δευτέρα έως και Κυριακή. Να μη γίνει έλεγχος εγκυρότητας.

Μονάδες 4

Δ3. Να καλεί για κάθε έναν από τους τηλεοπτικούς σταθμούς κατάλληλο υποπρόγραμμα, το οποίο να υπολογίζει και να επιστρέφει το μέσο πλήθος τηλεθεατών, που παρακολούθησαν το κεντρικό δελτίο ειδήσεών του, τη συγκεκριμένη εβδομάδα. Να αναπτύξετε το κατάλληλο υποπρόγραμμα.

Μονάδες 4

Δ4. Να εμφανίζει τα ονόματα των σταθμών για τους οποίους ο μέσος όρος τηλεθέασης του Σαββατοκύριακου (2 ημέρες) ήταν τουλάχιστον 10% μεγαλύτερος από το μέσο όρο τηλεθέασης στις καθημερινές (Δευτέρα έως και Παρασκευή).

Μονάδες 5

Δ5. Να εμφανίζει τα ονόματα των τηλεοπτικών σταθμών, οι οποίοι κάθε ημέρα, από Δευτέρα έως και Κυριακή, παρουσιάζουν συνεχώς, από ημέρα σε ημέρα, αύξηση τηλεθέασης. Αν δεν υπάρχουν τέτοιοι σταθμοί, να εμφανίζει το μήνυμα: «Κανένας σταθμός δεν είχε συνεχή αύξηση τηλεθέασης».

Μονάδες 5

```
# ----------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ----------------------------------------------------------------------
# Όνομα:        2012_1_D
# Περιγραφή:    Σφυγμομέτρηση τηλεθέασης εβδομάδας
# ΕνδΛύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ----------------------------------------------------------------------

# Ερώτημα Δ1, Δ2
names = []
views = []
for s in range(5):
    names.append(raw_input("Δώσε το όνομα του σταθμού = "))
    views.append([])
    for d in range(7):
        views[s].append(input("Δώσε τις ημερήσιες θεάσεις του σταθμού = "))


# Ερώτημα Δ3
def average_view(s):
    xsum = 0
```

```
    for d in range(7):
        xsum += views[s][d]
    return xsum * 1.0 / 7

average_views = []
for s in range(5):
    average_views.append(average_view(s))

# Ερώτημα Δ4
print "Σταθμοί με θέαση πάνω από 15% τα ΣΚ"
for s in range(5):
    sum_work = 0
    for d in range(5):
        sum_work += views[s][d]
    average_work = sum_work * 1.0 / 5

    sum_weekend = 0
    for d in range(5, 7):
        sum_weekend += views[s][d]
    average_weekend = sum_weekend * 1.0 / 2

    if average_weekend >= average_work * 1.10:
        print "Σταθμοί = ", names[s]

# Ερώτημα Δ5
print "Σταθμοί με αυξανόμενη θέαση "
count = 0
for s in range(5):
    found = True
    for d in range(6):
        if views[s][d+1] < views[s][d]:
            found = False
    if found:
        print names[s]
        count += 1
if count == 0:
    print "Δεν υπάρχει τέτοιος Σταθμός"
```

ΘΕΜΑ Γ_0_2013 (Μέτρηση SAR κινητών τηλεφώνων)

Η χρήση των κινητών τηλεφώνων, των φορητών υπολογιστών, των tablet υπολογιστών από τους νέους αυξάνεται ραγδαία. Ένας από τους στόχους των ερευνητών είναι να διερευνήσουν αν υπάρχουν επιπτώσεις στην υγεία των ανθρώπων από την αυξημένη έκθεση στα ηλεκτρομαγνητικά πεδία. Για τον σκοπό αυτό γίνονται μετρήσεις του ειδικού ρυθμού απορρόφησης (SAR) της ηλεκτρομαγνητικής ακτινοβολίας, πάνω στο ανθρώπινο σώμα. Ο δείκτης SAR μετράται σε Watt/Kgr και ο παγκόσμιος οργανισμός υγείας έχει θεσμοθετήσει ότι τα επιτρεπτά όρια για το κεφάλι και τον κορμό είναι μέχρι και 2 Watt/Kgr, ενώ για τα άκρα μέχρι και 4 Watt/Kgr. Θέλοντας να προσομοιάσουμε την έρευνα, θεωρούμε ότι σε 30 μαθητές έχουν τοποθετηθεί στον καθένα δυο μετρητές του δείκτη SAR, ο ένας στο κεφάλι και ο άλλος σε ένα από τα άνω άκρα, οι οποίοι καταγράφουν τις τιμές του αντίστοιχου δείκτη SAR κάθε 6 λεπτά.

Να αναπτύξετε αλγόριθμο σε ψευδογλώσσα, ο οποίος:

Γ1. Να διαβάζει τους πίνακες: ΚΩΔ[30], ο οποίος θα περιέχει τους κωδικούς των 30 μαθητών, τον πίνακα ΚΕΦ[30,10], του οποίου κάθε γραμμή θα αντιστοιχεί σε έναν μαθητή και θα έχει 10 τιμές που αντιστοιχούν στο SAR της κεφαλής για μια ώρα, καθώς και τον πίνακα ΑΚΡ[30,10] που κάθε γραμμή θα αντιστοιχεί σε έναν μαθητή και θα έχει 10 τιμές που αντιστοιχούν στο SAR του άκρου για μια ώρα.

Μονάδες 2

Γ2. Για κάθε μαθητή να καταχωρεί σε δισδιάστατο πίνακα ΜΟ[30,2] τις μέσες τιμές του SAR για το κεφάλι στην 1η στήλη και για το άκρο στη 2η στήλη.

Μονάδες 4

Γ3. Να εμφανίζει για κάθε μαθητή τον κωδικό του και ένα από τα μηνύματα, «Χαμηλός SAR», «Κοντά στα όρια», «Εκτός ορίων», όταν η μέση τιμή του SAR της κεφαλής, καθώς και η μέση τιμή του SAR ενός εκ των άκρων του κυμαίνονται στις παρακάτω περιοχές:

Μ.Ο. SAR κεφαλής	<=1,8	>1,8 και <=2	>2
Μ.Ο. SAR άκρου	<=3,6	>3,6 και <=4	>4
	Μήνυμα «Χαμηλός SAR»	«Κοντά στα όρια»	«Εκτός ορίων»

Το μήνυμα που θα εμφανίζεται θα πρέπει να είναι ένα μόνο για κάθε μαθητή και θα εξάγεται από τον συνδυασμό των τιμών των μέσων όρων των δυο SAR, όπου βαρύτητα θα έχει ο μέσος όρος, ο οποίος θα βρίσκεται σε μεγαλύτερη περιοχή τιμών. Για παράδειγμα, αν ο μέσος όρος SAR του άκρου έχει τιμή 3,8 και της κεφαλής έχει τιμή 1,5 τότε πρέπει να εμφανίζεται το μήνυμα «Κοντά στα όρια» και κανένα άλλο.

Μονάδες 7

Γ4. Θεωρώντας ότι όλες οι τιμές του πίνακα ΜΟ[30,2] είναι διαφορετικές, να εμφανίζει τις τρεις μεγαλύτερες τιμές για τον μέσο όρο SAR της κεφαλής και τους κωδικούς των μαθητών που αντιστοιχούν σε αυτές. Μετά να εμφανίζει τις τρεις μεγαλύτερες τιμές για τον μέσο όρο SAR του άκρου και τους κωδικούς των μαθητών που αντιστοιχούν σε αυτές.

Μονάδες 7

```
# ----------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ----------------------------------------------------------------------
# Όνομα:       2013_0_C
# Περιγραφή:   Μέτρηση SAR κινητών τηλεφώνων
# Ενδλύση:     TasosChatzipapadopoulos
# Python vs:   2.7.10
# Copyright:   (c) 2018
# Url:         http://users.sch.gr/chatzipap
# ----------------------------------------------------------------------

# Ερώτημα Γ1
code = []
head = []
lower = []
for i in range(30):
    code.append(raw_input("Δώσε τον κωδικό του μαθητή = "))
    head.append([])
    lower.append([])
    for j in range(10):
        head[i].append(input("Δώσε την μέτρηση στο κεφάλι = "))
        lower[i].append(input("Δώσε τη μέτρηση στα άκρα = "))

# Ερώτημα Γ2
average = []
for i in range(30):
    xsum_head = 0
    xsum_lower = 0
    for j in range(10):
        xsum_head += head[i][j]
        xsum_lower += lower[i][j]
    average.append([])
    average[i].append(xsum_head / 10.0)
    average[i].append(xsum_lower / 10.0)
print average

# Ερώτημα Γ3
for i in range(30):
    if average[i][0] > 2 or average[i][1] > 4:
        print "Εκτός Ορίων"
    elif average[i][0] > 1.8 or average[i][1] > 3.6:
        print "SAR κοντά στα όρια"
    else:
        print "Χαμηλός SAR"

# Ερώτημα Γ4
# Θα αποφύγουμε την προσέγγιση της λύσης με παράλληλη ταξινόμηση
# Δημιουργούμε 2 λίστες με τους ΜΟ στο κεφάλι και στα άκρα
average_head = [average[i][0] for i in range(30)]
average_lower = [average[i][1] for i in range(30)]

# Συνάρτηση η οποία επιστρέφει τους δείκτες των 3 μεγαλύτερων στοιχείων
# μίας λίστας ως λίστα 3 στοιχείων
def find_3maxs(a_list):
    imax3 = []
```

```
    for i in range(3):
    xmax = a_list[0]
    imax = 0
    for i in range(1, len(a_list)):
        if a_list[i] > xmax:
                if i not in imax3:
                    xmax = a_list[i]
                    imax = i
    imax3.append(imax)
    return imax3

# Δημιουργία λιστών με τους δείκτες των μεγαλύτερων τιμών
imax3_head = find_3maxs(average_head)
imax3_lower = find_3maxs(average_lower)

# Εμφάνιση αποτελεσμάτων
for i in range(3):
    print "Sar στο κεφάλι ", code[imax3_head[i]], average_head[imax3_head[i]],
    print "Sar στα άκρα", code[imax3_lower[i]], average_lower[imax3_lower[i]]
```

ΘΕΜΑ Δ_0_2013 (Ανταλλαγή μαθητών Comenius)

Σε ένα πρόγραμμα ανταλλαγής μαθητών Comenius συμμετέχουν μαθητές από δυο χώρες: Ελλάδα (EL) και Ισπανία (ES). Οι μαθητές αυτοί καλούνται να απαντήσουν σε μια ερώτηση όπου οι δυνατές απαντήσεις είναι:

1. Πολύ συχνά 2. Συχνά 3. Αρκετές φορές 4. Σπάνια 5. Ποτέ

Στην πρώτη φάση επεξεργασίας της ερώτησης πρέπει να καταγραφούν οι απαντήσεις από κάθε χώρα και να μετρήσουν για κάθε αριθμό απάντησης πόσες φορές υπάρχει, με σκοπό να αναφέρουν για κάθε χώρα, ποια απάντηση είχε τα μεγαλύτερα ποσοστά.

Για να βοηθήσετε στην επεξεργασία να αναπτύξετε πρόγραμμα σε ΓΛΩΣΣΑ το οποίο:

Δ1. α. Να περιέχει τμήμα δηλώσεων.

β. Να δημιουργεί δύο πίνακες EL[5] και ES[5] και να καταχωρίζει σε αυτούς την τιμή 0 σε όλα τα στοιχεία τους.

Μονάδες 2

Δ2. Για κάθε μαθητή να διαβάζει το όνομα της χώρας του και τον αριθμό της απάντησής του. Οι δυνατές τιμές για τη χώρα είναι: EL, ES και για την απάντηση 1,2,3,4,5. Η κάθε απάντηση θα πρέπει να προσμετράται σε έναν από τους δύο πίνακες EL[5], ES[5] ανάλογα με τη χώρα και στο αντίστοιχο στοιχείο. Δηλαδή, αν δοθούν για τιμές οι ES και 4, τότε θα πρέπει στο 4ο στοιχείο του πίνακα ES[5] να προστεθεί μια ακόμα καταχώριση. (Δεν απαιτείται έλεγχος εγκυρότητας τιμών)

Μονάδες 5

Δ3. Η προηγούμενη διαδικασία εισαγωγής δεδομένων και καταχώρισης απαντήσεων θα ελέγχεται από την ερώτηση «για Διακοπή της εισαγωγής πατήστε Δ ή δ», που θα εμφανίζεται, και ο χρήστης θα πρέπει να δώσει το χαρακτήρα Δ ή δ για να σταματήσει την επαναληπτική διαδικασία.

Μονάδες 3

Δ4. Στο τέλος για κάθε χώρα να εμφανίζει ποιος αριθμός απάντησης είχε το μεγαλύτερο ποσοστό, καθώς και το ποσοστό αυτό. Για την υλοποίηση αυτού του ερωτήματος θα χρησιμοποιήσετε δυο φορές το υποπρόγραμμα ΜΕΓ_ΠΟΣ που θα κατασκευάσετε στο ερώτημα Δ5. Θεωρούμε ότι για κάθε χώρα τα ποσοστά των απαντήσεων είναι διαφορετικά μεταξύ τους και δεν υπάρχει περίπτωση ισοβαθμίας.

Μονάδες 3

Δ5. Να αναπτύξετε το υποπρόγραμμα ΜΕΓ_ΠΟΣ το οποίο:

1. Να δέχεται έναν πίνακα ακεραίων 5 θέσεων.

2. Να βρίσκει το μεγαλύτερο στοιχείο του πίνακα και σε ποια θέση βρίσκεται.

3. Να βρίσκει το ποσοστό που κατέχει το μεγαλύτερο στοιχείο σε σχέση με το άθροισμα όλων των στοιχείων του πίνακα.

4. Να επιστρέφει στο κυρίως πρόγραμμα το ποσοστό αυτό, καθώς και την θέση στην οποία βρίσκεται.

Θεωρήστε ότι όλες οι τιμές των πινάκων είναι διαφορετικές και ότι για κάθε χώρα υπάρχει τουλάχιστον μια απάντηση στην ερώτηση.

Μονάδες 7

```
# ----------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ----------------------------------------------------------------------
# Όνομα:       2013_0_D
# Περιγραφή:   Ανταλλαγή μαθητών Comenius
# Ενδλύση:     TasosChatzipapadopoulos
# Python vs:   2.7.10
# Copyright:   (c) 2018
# Url:         http://users.sch.gr/chatzipap
# ----------------------------------------------------------------------

# Ερώτημα Δ1
el = [0] * 5
es = [0] * 5

# Ερώτημα Δ2
country = raw_input('Δώσε τη χώρα σου (EL)LADA, (ES)PANIA, EN(D)= The end =')
while country.upper() != 'D':
    a = input("Δώσε την απάντησή σου 1,2,3,4,5 = ")
    if country.upper() == "EL":
        el[a-1] += 1
    elif country.upper() == "ES":
        es[a-1] += 1
    # Ερώτημα Δ3
    country = raw_input('Δώσε τη χώρα σου (EL)LADA, (ES)PANIA, EN(D)= The end =')


# Ερώτημα Δ5
def max_pos(a_list):
    xmax = a_list[0]
    imax = 0
    for i in range(1, len(a_list)):
        if a_list[i] > xmax:
            xmax = a_list[i]
            imax = i
    xsum = 0
    xsum_no_max = 0
    for i in range(len(a_list)):
        xsum += a_list[i]
    max_percentage = xmax * 1.0 / xsum * 100
    return imax, max_percentage

# Ερώτημα Δ4
print "Ελλάδα ", max_pos(el)[0] + 1, max_pos(el)[1]
print "Ισπανία ", max_pos(es)[0] + 1, max_pos(es)[1]
```

ΘΕΜΑ Γ_1_2013 (Εκδήλωση συλλόγου γονέων & κηδεμόνων)

Ο σύλλογος γονέων και κηδεμόνων μιας περιοχής θέλει να διοργανώσει μια πολιτιστική εκδήλωση. Για το σκοπό αυτό, ζητά από κάθε σχολείο της περιοχής να προσφέρει κάποιο χρηματικό ποσό για την πραγματοποίησή της. Κάθε σχολείο έχει τη δυνατότητα να επικοινωνεί περισσότερες από μία φορές με το σύλλογο και να τροποποιεί την προσφορά του.

Να αναπτύξετε αλγόριθμο σε ψευδογλώσσα, ο οποίος:

Γ1. Να θεωρεί δεδομένο ένα πίνακα Σ[100] που περιέχει τα ονόματα των 100 σχολείων της περιοχής και να δημιουργεί πίνακα Π[100] που θα περιέχει τις αντίστοιχες χρηματικές προσφορές από κάθε σχολείο. Αρχικά να τοποθετηθεί σε κάθε στοιχείο του πίνακα Π[100] την τιμή -1.

Μονάδες 3

Γ2. α) Να διαβάζει το όνομα ενός σχολείου και να το αναζητά στον πίνακα Σ.

(μονάδες 4)

β) Να εμφανίζει το μήνυμα «Άγνωστο», όταν το σχολείο δε βρεθεί. Όταν το σχολείο βρεθεί, να σταματά την αναζήτηση, να διαβάζει τη χρηματική προσφορά του σχολείου και να την τοποθετεί στην αντίστοιχη θέση του πίνακα Π. (Όταν δοθεί η τιμή 0, σημαίνει ότι το σχολείο δεν μπορεί να προσφέρει χρήματα, δηλαδή έδωσε μηδενική προσφορά).Όταν δεν είναι η πρώτη φορά που δίνει προσφορά τότε να εμφανίζει το μήνυμα «ΤΡΟΠΟΠΟΙΗΣΗ ΠΡΟΣΦΟΡΑΣ» και να αντικαθιστά την προηγούμενη προσφορά του με τη νέα.

(μονάδες 6)

Μονάδες 10

Γ3. Να επαναλαμβάνει τις ενέργειες που περιγράφονται στο ερώτημα Γ2, μέχρις ότου όλα τα σχολεία να δώσουν τουλάχιστον μία προσφορά.

Μονάδες 3

Γ4. Να εμφανίζει:

α) το συνολικό χρηματικό ποσό που έχει συγκεντρωθεί,

β) το πλήθος των σχολείων που έδωσαν μηδενική προσφορά,

γ) το πλήθος των τροποποιήσεων που έγιναν στις προσφορές.

Μονάδες 4

```
# ----------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ----------------------------------------------------------------------
# Όνομα:        2013_1_C
# Περιγραφή:    Εκδήλωση συλλόγου γονέων & κηδεμόνων
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ----------------------------------------------------------------------

# Ερώτημα Γ1
schools = []
donations = []
for i in range(100):
    schools.append(raw_input("Δώσε το όνομα του σχολείου = "))
    donations.append(-1)

# Ερώτημα Γ3
modifications_count = 0
donations_count = 0
while donations_count < 100:
    # Ερώτημα Γ2
    i = 0
    found = False
    school = raw_input("Δώσε το όνομα σχολείου για αναζήτηση = ")
    while i <= 100 and not found:
        if schools[i] == school:
            found = True
            pos = i
        else:
            i += 1

    if not found:
        print "Το Σχολείο ", school, "δε βρέθηκε"
    else:
        donate = input("Πόσα χρήματα = ")
        if donations[pos] != -1:
            print "Τροποποίηση..."
            modifications_count += 1
        else:
            donations_count += 1
        donations[pos] = donate

# Ερώτημα Γ4
sum_donations = 0
zero_count = 0
for i in range(100):
    sum_donations += donations[i]
    if donations[i] == 0:
        zero_count += 1

print "Σύνολο δωρεών = ", sum_donations
print "Μηδενικές δωρεές = ", zero_count
print "Τροποποιήσεις = ", modifications_count
```

ΘΕΜΑ Δ_1_2013 (Μετάδοση και λήψη bits έλεγχος λαθών)

Τα δεδομένα (κείμενο, εικόνα, ήχος, κλπ), κατά τη μετάδοσή τους μέσω ενσύρματων ή ασύρματων καναλιών επικοινωνίας, αλλοιώνονται λόγω του θορύβου που χαρακτηρίζει κάθε κανάλι. Ο τρόπος προστασίας των δεδομένων μετάδοσης είναι ο ακόλουθος:

Για κάθε bit (ακέραιος με τιμή 0 ή 1), που ο πομπός θέλει να στείλει, μεταδίδει μια λέξη, που αντιστοιχεί σε πίνακα ΜΕΤΑΔΟΣΗ[31] με όλες τις τιμές του ταυτόσημες με το προς μετάδοση bit, δηλαδή, αν πρόκειται να σταλεί το bit 1, τότε η λέξη που μεταδίδεται είναι η 11...1 μήκους 31 bits, ενώ αν πρόκειται να σταλεί το bit 0, τότε η λέξη που μεταδίδεται είναι η 00...0, μήκους 31 bits. Ο δέκτης λαμβάνει λέξη μήκους 31 bits, τα οποία τοποθετούνται σε πίνακα ΛΗΨΗ[31]. Έχουμε «ΛΑΝΘΑΣΜΕΝΗ ΛΗΨΗ», εάν υπάρχει τουλάχιστον ένα στοιχείο του πίνακα ΛΗΨΗ[31] με διαφορετική τιμή από αυτήν του αντίστοιχου στοιχείου του πίνακα ΜΕΤΑΔΟΣΗ[31]. Εάν το πλήθος των 1 του πίνακα ΛΗΨΗ[31] είναι μεγαλύτερο από το πλήθος των 0, τότε ο δέκτης αποφασίζει ότι ο πομπός έστειλε 1, ενώ σε αντίθετη περίπτωση ο δέκτης αποφασίζει ότι ο πομπός έστειλε 0. Σε κάθε περίπτωση, αν περισσότερα από τα μισά των 31 bits της λέξης μετάδοσης έχουν αλλοιωθεί, τότε ο δέκτης θα έχει πάρει «ΛΑΝΘΑΣΜΕΝΗ ΑΠΟΦΑΣΗ».

Να γραφεί πρόγραμμα σε ΓΛΩΣΣΑ, το οποίο να κάνει τα εξής:

Δ1. Να περιλαμβάνει κατάλληλο τμήμα δηλώσεων.

Μονάδες 3

Δ2. Για κάθε τιμή ποιότητας του καναλιού, που χαρακτηρίζεται από ακεραίους από 1 έως και 10, να πραγματοποιούνται το πολύ 100.000 διαφορετικές προσπάθειες μετάδοσης-λήψης και διόρθωσης λαθών. Εάν όμως ληφθούν 100 λανθασμένες αποφάσεις, τότε να διακόπτεται η διαδικασία για τη συγκεκριμένη τιμή ποιότητας του καναλιού.

Μονάδες 4

Δ3.Σε κάθε προσπάθεια μετάδοσης-λήψης και διόρθωσης λαθών να πραγματοποιούνται οι ακόλουθες ενέργειες:

α. Να διαβάζει (χωρίς έλεγχο εγκυρότητας των τιμών τους) τη μεταδοθείσα λέξη, καθώς και τη ληφθείσα λέξη και να ελέγχει, εάν αυτές ταυτίζονται.

β. Να διορθώνει τη ληφθείσα λέξη στο δέκτη, βάσει της παραπάνω περιγραφής του αλγορίθμου.

Μονάδες 9

Δ4. α. Να αποθηκεύει, για κάθε τιμή ποιότητας καναλιού, σε πίνακα ΛΑΘΗΑΠΟΦ[10] το ποσοστό των λανθασμένων αποφάσεων και σε πίνακα ΛΑΘΗΛΗΨ[10] το ποσοστό των λανθασμένων λήψεων.

β. Να εμφανίζει συγκεντρωτικά τα ποσοστά των λανθασμένων αποφάσεων και λανθασμένων λήψεων στο δέκτη.

Μονάδες 4

```
# ---------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ---------------------------------------------------------------------
# Όνομα:        2013_1_D
# Περιγραφή:    Μετάδοση και λήψη bits έλεγχος λαθών
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ---------------------------------------------------------------------

# Ερώτημα Δ1
send = receive = []
wrong_decisions_list = faults_receive_list = []
for i in range(10):
    wrong_decisions = 0
    tries = 0
    faults = 0
    while tries <= 100000 and wrong_decisions <= 100:
        ok_bits = 0
        zeros = 0
        ones = 0
        for bit in range(31):
            send.append(input("Δώσε bit αποστολής = "))
            receive.append(input("Δώσε bit λήψης ="))

            if send[bit] == receive[bit]:
                ok_bits += 1
            else:
                if receive[bit] == 0:
                    zeros += 1
                elif receive[bit] == 1:
                    ones += 1

        if ok_bits < 31:
            faults += 1
            if zeros > ones:
                prevail = 0
            else:
                prevail = 1
            receive = [prevail for bit in range(31)]

            if ok_bits <= 15:
                wrong_decisions += 1

        tries += 1
    w = wrong_decisions * 1.0 / tries * 100
    f = faults * 1.0 / tries * 100
    wrong_decisions_list.append(w)
    faults_receive_list.append(f)
    print i, w, f
```

ΘΕΜΑ Γ_0_2014 (Αγορές προϊόντων μετρητοίς ή με δόσεις)

Ένας πελάτης αγοράζει προϊόντα από ένα κατάστημα. Να αναπτύξετε αλγόριθμο ο οποίος:

Γ1. Για κάθε προϊόν που αγοράζει ο πελάτης, να διαβάζει τον κωδικό του, τον αριθμό τεμαχίων που αγοράστηκαν και την τιμή τεμαχίου. Η διαδικασία ανάγνωσης να σταματά, όταν δοθεί ως κωδικός ο αριθμός 0.

Μονάδες 3

Γ2. Αν ο λογαριασμός δεν υπερβαίνει τα 500 ευρώ, να εμφανίζει το μήνυμα «ΠΛΗΡΩΜΗ ΜΕΤΡΗΤΟΙΣ». Διαφορετικά, να υπολογίζει και να εμφανίζει το πλήθος των απαιτούμενων για την εξόφληση δόσεων, όταν η εξόφληση γίνεται με άτοκες μηνιαίες δόσεις, ως εξής: Τον πρώτο μήνα η δόση θα είναι 20 ευρώ και κάθε επόμενο μήνα θα αυξάνεται κατά 5 ευρώ, μέχρι να εξοφληθεί το συνολικό ποσό.

Μονάδες 6

Γ3. Να υπολογίζει και να εμφανίζει τον συνολικό αριθμό των τεμαχίων με τιμή τεμαχίου μεγαλύτερη των 10 ευρώ.

Μονάδες 5

Γ4. Να υπολογίζει και να εμφανίζει τον συνολικό αριθμό των τεμαχίων με τη μέγιστη τιμή τεμαχίου.

Μονάδες 6

```
# ------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------
# Όνομα:        2014_0_C
# Περιγραφή:    Αγορές προϊόντων μετρητοίς ή με δόσεις
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ------------------------------------------------------------------

# Ερώτημα Γ1
code = raw_input("Δώσε τον κωδικό του προϊόντος = ")
total = 0
total_items = 0
max_price = 0
sum_max_items = 0
while code != '0':
    items = input("Πόσα τεμάχια = ")
    price = input("Δώσε την τιμή του προϊόντος = ")
    total = total + items * price

    # Ερώτημα Γ3
    if price > 10:
        total_items += items

    # Ερώτημα Γ4
    if price > max_price:
        max_price = price
        sum_max_items = items
    elif price == max_price:
        sum_max_items += items
```

```
    print total
    code = raw_input("Δώσε τον κωδικό του προϊόντος = ")

# Ερώτημα Γ2
if total <= 500:
    print "ΠΛΗΡΩΜΗ ΜΕΤΡΗΤΟΙΣ"
else:
    print "ΠΛΗΡΩΜΗ ΜΕ ΔΟΣΕΙΣ"
    count = 1
    dose = 20
    pay = 20
    while pay < total:
        # Αλλάζουμε λίγο τα αποτελέσματα και κάνουμε εμφανίζουμε
        # τις δόσεις με κάθε προιόν
        print count, dose, pay
        if dose <= total - pay:
            dose += 5
            # print dose, pay

        else:
            dose = total - pay
            # print dose, pay
        pay += dose
        count += 1
    print count, dose, pay
    print "ΔΟΣΕΙΣ = ", count

print "Συνολικός αριθμός των τεμαχίων με τιμή τεμαχίου μεγαλύτερη των 10 ευρώ"
print total_items
print "Συνολικός αριθμός των τεμαχίων με τη μέγιστη τιμή τεμαχίου"
print sum_max_items
```

ΘΕΜΑ Δ_0_2014 (Στατιστικά Επισκέψεων Ιστοτόπων)

Μια εταιρεία Πληροφορικής καταγράφει, για δέκα ιστότοπους, τον αριθμό των επισκέψεων που δέχεται ο καθένας, κάθε μέρα, για τέσσερις εβδομάδες.

Να αναπτύξετε αλγόριθμο, ο οποίος:

Δ1. Για καθένα από τους ιστότοπους να διαβάζει το όνομά του και τον αριθμό των επισκέψεων που δέχθηκε ο ιστότοπος για καθεμιά ημέρα. Δεν απαιτείται έλεγχος εγκυρότητας τιμών.

Μονάδες 2

Δ2. Να εμφανίζει το όνομα κάθε ιστοτόπου και τον συνολικό αριθμό των επισκέψεων που δέχθηκε αυτός στο διάστημα των τεσσάρων εβδομάδων.

Μονάδες 3

Δ3. Να εμφανίζει τα ονόματα των ιστοτόπων που κάθε μέρα στο διάστημα των τεσσάρων εβδομάδων δέχθηκαν περισσότερες από 500 επισκέψεις. Αν δεν υπάρχουν τέτοιοι ιστότοποι, να εμφανίζει κατάλληλο μήνυμα.

Μονάδες 6

Δ4. Να διαβάζει το όνομα ενός ιστοτόπου. Αν το όνομα αυτό δεν είναι ένα από τα δέκα ονόματα που έχουν δοθεί, να το ξαναζητά, μέχρι να δοθεί ένα από αυτά τα ονόματα. Να εμφανίζει τους αριθμούς των εβδομάδων (1-4) κατά τη διάρκεια των οποίων ο συνολικός (εβδομαδιαίος) αριθμός επισκέψεων στον ιστότοπο αυτό είχε τη μέγιστη τιμή.

Μονάδες 9

```
# ----------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ----------------------------------------------------------------------
# Όνομα:       2014_0_D
# Περιγραφή:   Στατιστικά Επισκέψεων Ιστοτόπων
# Ενδλύση:     TasosChatzipapadopoulos
# Python vs:   2.7.10
# Copyright:   (c) 2018
# Url:         http://users.sch.gr/chatzipap
# ----------------------------------------------------------------------

# Ερώτημα Δ1
www = []
hits = []
for w in range(10):
    www.append(raw_input("Δώσε το όνομα του ιστότοπου = "))
    hits.append([])
    for d in range(28):
        hits[w].append(input("Δώσε τις επισκέψεις του ιστότοπου = "))

# Ερώτημα Δ2
total_hits = []
for w in range(10):
    xsum = 0
    for d in range(28):
        xsum += hits[w][d]
    print www[w], xsum

# Ερώτημα Δ3
```

```
count = 0
print "Ονόματα ιστοτόπων με ημερήσια κίνηση πάνω από 500 επισκέψεις"
for w in range(10):
    flag = True
    for d in range(28):
        if hits[w][d] < 500:
            flag = False
    if flag == True:
        count += 1
        print www[w]
if count == 0:
    print "Δυστυχώς δε βρέθηκε κανένας ιστότοπος"


# Ερώτημα Δ4
# Συνάρτηση εύρεσης ονόματος
def search(url):
    found = -1
    for w in range(10):
        if url == www[w]:
            found = w
    return found

# Εισαγωγή υπαρκτού ονόματος url
url = raw_input("Δώσε το όνομα του ιστότοπου = ")
while search(url) == -1:
    url = raw_input("Δώσε το όνομα του ιστότοπου = ")
pos = search(url)

# Εύρεση εβδομαδιαίων αθροισμάτων επισκέψεων
weekly_hits = []
xsum = 0
for d in range(28):
    if d % 7 != 0 or d == 0:
        xsum += hits[pos][d]
    else:
        weekly_hits.append(xsum)
        xsum = 0
weekly_hits.append(xsum)

# Εϋρεση μεγαλύτερης τιμής
xmax = weekly_hits[0]
for i in range(4):
    if weekly_hits[i] > xmax:
        xmax = weekly_hits[i]

# Εύρεση αριθμών εβδομαδων με τη μεγαλύτερη επισκεψιμότητα
for i in range(4):
    if weekly_hits[i] == xmax:
        print i + 1
```

ΘΕΜΑ Γ_1_2014 (Επίλυση Εξίσωσης Α·x+Β·y+Γ·z=Δ)

Δίνεται η εξίσωση Α·x+Β·y+Γ·z=Δ. Να αναπτύξετε αλγόριθμο, ο οποίος, θεωρώντας δεδομένες τις τιμές των Α, Β, Γ και Δ:

Γ1. Να εμφανίζει όλες τις λύσεις (τριάδες) της εξίσωσης, εξετάζοντας όλους τους δυνατούς συνδυασμούς ακεραίων τιμών των x, y, z, που είναι μεγαλύτερες από -100 και μικρότερες από 100. Αν δεν υπάρχουν τέτοιες λύσεις, να εμφανίζει κατάλληλο μήνυμα.

Μονάδες 8

Εφόσον υπάρχουν τέτοιες λύσεις:

Γ2. Να εμφανίζει την πρώτη λύση (τριάδα) για την οποία το άθροισμα των x, y, z έχει τη μεγαλύτερη τιμή.

Μονάδες 4

Γ3. Να εμφανίζει το πλήθος των λύσεων της εξίσωσης για τις οποίες τα x, y, z είναι θετικοί άρτιοι αριθμοί.

Μονάδες 4

Γ4. Να εμφανίζει το ποσοστό των λύσεων της εξίσωσης για τις οποίες ένα μόνο από τα x, y, z είναι ίσο με μηδέν.

Μονάδες 4

```
# ----------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ----------------------------------------------------------------------
# Όνομα:       2014_1_C
# Περιγραφή:   Επίλυση Εξίσωσης Α·x+Β·y+Γ·z=Δ
# Ενδλύση:     TasosChatzipapadopoulos
# Python vs:   2.7.10
# Copyright:   (c) 2018
# Url:         http://users.sch.gr/chatzipap
# ----------------------------------------------------------------------

# Ερώτημα Γ1
[a, b, c, d] = input("Δώσε τα a,b,c,d στην εξίσωση Α·x+Β·y+Γ·z=Δ ")
a = float(a)
b = float(b)
c = float(c)
d = float(d)
print a, b, c, d
solutions = 0
max_sum = -300
positives = 0
zeros = 0
for x in range(-99, 100):
    for y in range(-99, 100):
        for z in range(-99, 100):

            # Γ1
            fx = a*x + b*y + c*z - d
            if fx == 0:
                print x, y, z
                solutions += 1

                # Γ2
```

```
                xsum = x + y + z
                if xsum > max_sum:
                    max_sum = xsum
                    max_nums = x, y, z

                # Γ3
                if (x > 0 and y > 0 and z > 0
                    and
                    x % 2 == 0 and y % 2 == 0 and z % 2 == 0):
                    positives += 1
                    print "---", x, y, z

                # Γ4
                if (x == 0 and y*z != 0
                   or
                    y == 0 and x*z != 0
                    or
                    z == 0 and x*y != 0):
                        zeros += 1
                        print "----", x, y, z

if solutions == 0:
    print "Δεν υπάρχουν λύσεις στο διάστημα -100 έως 100"
else:
    print "την πρώτη λύση (τριάδα) για την οποία το άθροισμα των x, y, z έχει τη
μεγαλύτερη τιμή"
    print max_nums

    print "το πλήθος των λύσεων της εξίσωσης για τις οποίες τα x, y, z είναι θετικοί
άρτιοι αριθμοί"
    print positives

    print "το ποσοστό των λύσεων της εξίσωσης για τις οποίες ένα μόνο από τα x, y, z
είναι ίσο με μηδέν"
    print zeros * 1.0 / solutions * 100, "%"
```

ΘΕΜΑ Δ_1_2014 (Αποτελέσματα σταυρών δημοτικών εκλογών)

Στις πρόσφατες δημοτικές εκλογές, σε κάποιο δήμο της χώρας, χρησιμοποιήθηκαν για την ψηφοφορία 217 αίθουσες (εκλογικά τμήματα), σε 34 δημόσια κτήρια (εκλογικά καταστήματα). Τα τμήματα αριθμήθηκαν με τη σειρά, από τό 1 μέχρι το 217, έτσι ώστε οι αριθμοί των εκλογικών τμημάτων κάθε καταστήματος να είναι διαδοχικοί: αριθμήθηκαν πρώτα τα τμήματα του πρώτου καταστήματος, στη συνέχεια τα τμήματα του δεύτερου καταστήματος κ.ο.κ. Το ψηφοδέλτιο ενός από τους συμμετέχοντες συνδυασμούς είχε 65 υποψηφίους. Κάθε ψηφοφόρος ψηφίζει σημειώνοντας σταυρό δίπλα στο όνομα κάθε υποψηφίου που επιλέγει.

Να αναπτύξετε αλγόριθμο, ο οποίος:

Δ1. Να διαβάζει:

α. Το πλήθος των εκλογικών τμημάτων για κάθε εκλογικό κατάστημα. Να γίνεται έλεγχος εγκυρότητας των τιμών που δίνονται, ώστε αυτές να είναι θετικές και το άθροισμά τους να είναι ίσο με 217. (μονάδες 4)

β. Τα ονόματα των υποψηφίων του συνδυασμού. (μονάδα 1)

γ. Τον αριθμό των σταυρών που έλαβε καθένας από τους 65 υποψηφίους του συνδυασμού, σε κάθε εκλογικό τμήμα. (μονάδα 1)

Μονάδες 6

Δ2. Να εμφανίζει τον συνολικό αριθμό σταυρών που έλαβε κάθε υποψήφιος.

Μονάδες 2

Δ3. Να εμφανίζει τα ονόματα των υποψηφίων που έλαβαν τους περισσότερους συνολικούς σταυρούς στο δεύτερο εκλογικό κατάστημα.

Μονάδες 5

Δ4. Να εμφανίζει, σε αλφαβητική σειρά, τα ονόματα των δέκα πρώτων σε σταυρούς υποψηφίων. Σε περίπτωση που υπάρχουν υποψήφιοι που έλαβαν τον ίδιο συνολικό αριθμό σταυρών με τον δέκατο, να εμφανίζει και τα δικά τους ονόματα.

Μονάδες 7

```
# ----------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ----------------------------------------------------------------------
# Όνομα:       2014_1_D
# Περιγραφή:   Αποτελέσματα σταυρών δημοτικών εκλογών
# Ενδλύση:     TasosChatzipapadopoulos
# Python vs:   2.7.10
# Copyright:   (c) 2018
# Url:         http://users.sch.gr/chatzipap
# ----------------------------------------------------------------------

# Ερώτημα Δ1,α
centers = []
sum_sectors = 0
while sum_sectors != 217:
    for i in range(34):
        sectors = input("Πλήθος των εκλογικών τμημάτων του καταστήματος = ")
        while sectors < 0:
           sectors = input("Πλήθος των εκλογικών τμημάτων του καταστήματος = ")
        centers.append(sectors)
        sum_sectors += sectors
    if sum_sectors != 217:
        print "Λάθος άθροισμα τμημάτων ξανακάνετε εισαγωγή"
        sum_sectors = 0
        cemters = []

# Ερώτημα Δ1,β
names = []
for n in range(65):
    names.append(raw_input("Δώσε όνομα υποψηφίου = "))

# Ερώτημα Δ1,γ
votes = []
for n in range(65):
    votes.append([])
    for s in range(217):
        votes[n].append(input("Δώσε πλήθος σταυρών = "))

# Ερώτημα Δ2
sum_votes = []
for n in range(65):
    vsum = 0
    for s in range(217):
        vsum += votes[n][s]
    sum_votes.append(vsum)

for n in range(65):
    print names[n], sum_votes[n]

# Ερώτημα Δ3
sum_votes2 = []
for n in range(65):
    vsum2 = 0
    for s in range(sectors[0], sectors[0] + sectors[1] + 1):
        vsum2 += votes[n][s]
```

```
    sum_votes2.append(vsum2)

smax = sum_votes2[0]
for n in range(1,65):
    if sum_votes2[n] > smax:
        smax = sum_votes2[n]

for n in range(65):
    if sum_votes2[n] == smax:
        print names[n]

# Ερώτημα Δ4

for i in range(64):
    for j in range(64, i, -1):
        if sum_votes[j] > sum_votes[j-1]:
            sum_votes[j], sum_votes[j-1] = sum_votes[j-1], sum_votes[j]
            names[j], names[j-1] = names[j-1], names[j]

# Εύρεση ισοψηφισάντων
key = sum_votes[9]
flag = False
i = 10
count = 0
while not flag:
    if key == sum_votes[i]:
        count += 1
    else:
        flag = True

limit = 10 + count
for i in range(limit - 1):
    for j in range(limit -1, i, -1):
        if names[j] < names[j-1]:
            sum_votes[j], sum_votes[j-1] = sum_votes[j-1], sum_votes[j]
            names[j], names[j-1] = names[j-1], names[j]

for i in range(limit):
    print names[i], votes[i]
```

ΘΕΜΑ Γ_0_2015 (Τοποθέτηση δεμάτων σε αποθήκες αεροδρομίου)

Μία εταιρεία μεταφοράς δεμάτων διαθέτει δύο αποθήκες, Α και Β, στο αεροδρόμιο. Κατά την παραλαβή δεμάτων, κάθε δέμα τοποθετείται στην αποθήκη που έχει εκείνη τη στιγμή τον περισσότερο ελεύθερο χώρο. Αν ο ελεύθερος χώρος της αποθήκης Α είναι ίσος με τον ελεύθερο χώρο της αποθήκης Β, το δέμα τοποθετείται στην αποθήκη Α. Όταν όμως το δέμα δεν χωρά σε καμία από τις δύο αποθήκες, προωθείται στις κεντρικές εγκαταστάσεις της εταιρείας, που βρίσκονται εκτός αεροδρομίου.

Γ1. Να κατασκευάσετε πρόγραμμα που:

α. Να περιλαμβάνει κατάλληλο τμήμα δηλώσεων. (μονάδες 2)

β. Να διαβάζει τα μεγέθη ελεύθερου χώρου των αποθηκών Α και Β.

(μονάδες 2)

γ. Να διαβάζει το μέγεθος κάθε εισερχόμενου δέματος και να εμφανίζει το όνομα της αποθήκης (Α ή Β) στην οποία θα τοποθετηθεί αυτό ή να εμφανίζει το μήνυμα «Προώθηση», όταν το δέμα δεν χωρά σε καμία από τις αποθήκες Α ή Β. Η διαδικασία παραλαβής τερματίζεται, όταν εισαχθεί ως μέγεθος δέματος η τιμή 0. (μονάδες 6)

δ. Στη συνέχεια, να καλεί υποπρόγραμμα, το οποίο να βρίσκει και να εμφανίζει το όνομα της αποθήκης (Α ή Β) στην οποία τοποθετήθηκαν τα περισσότερα δέματα, ή το μήνυμα «Ισάριθμα» σε περίπτωση που στις δύο αποθήκες Α και Β τοποθετήθηκαν ισάριθμα δέματα, ή το μήνυμα «Καμία αποθήκευση στο αεροδρόμιο», αν κανένα δέμα δεν τοποθετήθηκε σε οποιαδήποτε από τις αποθήκες Α ή Β. (μονάδες 2)

Μονάδες 12

Γ2. Να κατασκευάσετε το υποπρόγραμμα που περιγράφεται στο ερώτημα Γ1.δ.

Μονάδες 8

```
# ----------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ----------------------------------------------------------------------
# Όνομα:        2015_0_C
# Περιγραφή:    Τοποθέτηση δεμάτων σε αποθήκες αεροδρομίου
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ----------------------------------------------------------------------


# Γ2
def count(a, b):
    if a > b:
        return "Αποθήκη Α"
    elif b > a:
        return "Αποθήκη Β"
    elif a == b and a*b != 0:
        return "Ισάριθμες αποθήκες Α & Β"
    else:
        return "Καμία αποθήκευση στις αποθήκες Α & Β"

# Γ1,β
a_storage = input("Δώσε τον ελεύθερο χώρο της αποθήκης Α = ")
b_storage = input("Δώσε τον ελεύθερο χώρο της αποθήκης Β = ")
a_count = b_count = 0

size = input("Δώσε το μέγεθος του δέματος ή 0 για διακοπή = ")
while size != 0:
    # Γ1,γ
    if a_storage >= b_storage and a_storage >= size:
        a_storage -= size
        a_count += 1
    elif a_storage < b_storage and b_storage >= size:
        b_storage -= size
        b_count += 1
    else:
        print "Προώθηση"
    size = input("Δώσε το μέγεθος του δέματος ή 0 για διακοπή = ")

# Γ1,δ
print count(a_count, b_count)
```

ΘΕΜΑ Δ_0_2015 (Βαθμολογίες διαγωνισμού τραγουδιού)

Ένας διαγωνισμός τραγουδιού διεξάγεται σε δύο φάσεις. Στην πρώτη φάση γίνεται ακρόαση των 45 τραγουδιών που διαγωνίζονται και κάθε μέλος της επταμελούς κριτικής επιτροπής βαθμολογεί το κάθε τραγούδι με βαθμό από 1 έως 10. Στη δεύτερη φάση προκρίνεται κάθε τραγούδι που συγκέντρωσε συνολική βαθμολογία μεγαλύτερη του 50 και το οποίο όλοι οι κριτές έχουν βαθμολογήσει τουλάχιστον με 5.

Να γραφεί αλγόριθμος, ο οποίος:

Δ1. Για κάθε τραγούδι να διαβάζει τον τίτλο του και τον βαθμό που έδωσε κάθε κριτής. Δεν απαιτείται έλεγχος εγκυρότητας.

Μονάδες 3

Δ2. Να υπολογίζει και να εμφανίζει τη συνολική βαθμολογία του κάθε τραγουδιού, η οποία προκύπτει ως το άθροισμα των βαθμών όλων των κριτών.

Μονάδες 2

Δ3. Να βρίσκει και να εμφανίζει τους τίτλους των τραγουδιών που προκρίνονται στη δεύτερη φάση του διαγωνισμού. Αν κανένα τραγούδι δεν προκρίνεται στη δεύτερη φάση, να εμφανίζει κατάλληλο μήνυμα.

Μονάδες 6

Δ4. Να βρίσκει και να εμφανίζει το πλήθος των κριτών που έδωσαν τον μέγιστο βαθμό τους σε ένα μόνο τραγούδι.

Μονάδες 9

```
# ---------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ---------------------------------------------------------------------
# Όνομα:        2015_0_D
# Περιγραφή:    Βαθμολογίες διαγωνισμού τραγουδιού
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ---------------------------------------------------------------------

# Δ1
titles = []
grades = []
for t in range(45):
    titles.append(raw_input("Δώσε τον τίτλο του τραγουδιού = "))
    grades.append([])
    for g in range(7):
        grades[t].append(input("Δώστε το βαθμό του τραγουδιού = "))

# Δ2
sum_grades = []
for t in range(45):
    gsum = 0
    for g in range(7):
        gsum += grades[t][g]
        sum_grades.append(gsum)
    print titles[t], sum_grades[t]
```

```
# Δ3
count = 0
song_pass = False
print "Τραγούδια που προκρίθηκαν στη 2η φάση ..."
for t in range(45):
    if sum_grades[t] >= 50:
        song_pass = True
        for g in range(7):
            if grades[t][g] < 5:
                song_pass = False
    if song_pass:
        count += 1
        print titles[t]
if count == 0:
    print "Δεν προκρίθηκε κανένα τραγούδι"

# Δ4
# Εύρεση του μεγαλύτερου βαθμού κάθε κριτή και εκχώρηση του στη λίστα:
g_max = []
for g in range(7):
    gmax = 0
    for t in range(45):
        if grades[t][g] > gmax:
            gmax = grades[t][g]
    g_max.append(gmax)

# Αναζήτηση του μεγαλύτερου βαθμού καθε κριτή στις βαθμολογίες τους
judges = 0
for gmax in g_max:
    count = 0
    for g in range(7):
        for t in range(45):
            if gmax == grades[t][g]:
                count += 1
    if count == 1:
        judges += 1

print "το πλήθος των κριτών που έδωσαν τον\
μέγιστο βαθμό τους σε ένα μόνο τραγούδι ="
print judges
```

ΘΕΜΑ Γ_1_2015 (Ονοματολογία χημικών ενώσεων)

Σύμφωνα με το διεθνές σύστημα ονοματολογίας της IUPAC, το όνομα ενός άκυκλου υδρογονάνθρακα CxHy με ευθύγραμμη ανθρακική αλυσίδα αποτελείται από τρία συνθετικά. Το πρώτο συνθετικό (σ1) καθορίζεται από τον αριθμό x των ατόμων άνθρακα, ως εξής: Όταν x=1, η τιμή του σ1 είναι μεθ· όταν x=2, η τιμή του σ1 είναι αιθ· όταν x=3, η τιμή του σ1 είναι προπ· όταν x=4, η τιμή του σ1 είναι βουτ· όταν x=5, η τιμή του σ1 είναι πεντ· όταν x=6, η τιμή του σ1 είναι εξ κ.ο.κ. Το δεύτερο συνθετικό (σ2) εξαρτάται από τον αριθμό x των ατόμων του άνθρακα και από τον αριθμό y των ατόμων υδρογόνου και η τιμή του είναι σ2=άν ή σ2=έν ή σ2=ίν ή σ2=αδιέν, σύμφωνα με τις συνθήκες που φαίνονται στον Πίνακα ΙΙ.

Τιμή του σ2	Συνθήκη
άν y=2x+2,	x≥1
έν y=2x,	x≥2
ίν y=2x-2,	x≥2
αδιέν y=2x-2,	x≥3

Πίνακας ΙΙ

Το τρίτο συνθετικό (σ3) είναι σε κάθε περίπτωση η κατάληξη ιο. Όπως φαίνεται στον Πίνακα ΙΙ, όταν x≥3, η τιμή του σ2 είναι ίν ή αδιέν. Ο τρόπος καθορισμού του ορθού ονόματος της ένωσης στην περίπτωση αυτή δεν μας ενδιαφέρει στο πλαίσιο της άσκησης.

Για παράδειγμα, όταν x=3 και y=8, η ένωση είναι το προπ-άν-ιο, ενώ αν x=3 και y=4, η ένωση είναι το προπ-ίν-ιο ή το προπ-αδιέν-ιο.

Να κατασκευάσετε αλγόριθμο ο οποίος:

Γ1. Να ζητάει τον αριθμό ατόμων άνθρακα της χημικής ένωσης, κάνοντας έλεγχο εγκυρότητας ώστε αυτός να είναι θετικός.

Μονάδες 2

Γ2. Να ζητάει τον αριθμό ατόμων υδρογόνου της χημικής ένωσης, κάνοντας έλεγχο εγκυρότητας ώστε να ικανοποιείται τουλάχιστον μία από τις συνθήκες του Πίνακα ΙΙ.

Μονάδες 6

Γ3. Να εκχωρεί στις μεταβλητές

σ1: το πρώτο συνθετικό του ονόματος της χημικής ένωσης. Θεωρείστε ότι δίνεται πίνακας Π, σε διαδοχικές θέσεις του οποίου βρίσκονται ήδη καταχωρισμένα τα λεκτικά που αντιστοιχούν στον αριθμό των ατόμων του άνθρακα (μονάδες 2) και

σ3: την κατάληξη του ονόματος της χημικής ένωσης (μονάδες 2).

Μονάδες 4

Γ4. Να υπολογίζει το σ2 και να εμφανίζει το όνομα (ή τα ονόματα) της χημικής ένωσης, εμφανίζοντας τα τρία συνθετικά, το ένα δίπλα στο άλλο, χωρισμένα με το χαρακτήρα «–».

Μονάδες 8

```
# ---------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ---------------------------------------------------------------------
# Όνομα:       2015_1_C
```

```
# Περιγραφή:   Ονοματολογία χημικών ενώσεων
# Ενδλύση:     TasosChatzipapadopoulos
# Python vs:   2.7.10
# Copyright:   (c) 2018
# Url:         http://users.sch.gr/chatzipap
# ------------------------------------------------------------------------

# Γ1
x = input("Δώσε το πλήθος των ατόμων του άνθρακα = ")
while x <= 0:
    print "Λάθος Δεδομένα"
    x = input("Δώσε το πλήθος των ατόμων του άνθρακα = ")


# Γ2
def check(x, y):
    check_y = False
    if y >= 1 and y % 2 == 0:
        check_y = True
    return check_y

y = input("Δώσε το πλήθος των ατόμων του υδρογόνου = ")
while not check(x, y):
    print "Λάθος Δεδομένα.."
    y = input("Δώσε το πλήθος των ατόμων του υδρογόνου = ")
    check(x, y)

# Γ3
def s1(x):
    x -= 1
    s = ["μεθ", "αιθ", "προπ",
         "βουτ", "πεντ", "εξ"]
    return s[x]

def s3():
    return 'ιο'

# Γ4
def s2(x, y):
    s = ''
    if y == 2*x + 2:
        if x >= 1:
            s = 'άν'
    if y == 2*x:
        if x >= 2:
            s = 'έν'
    if y == 2*x - 2:
        if x >= 2:
            s = 'ίν'
        elif x >= 3:
            s = 'αδιέν'
    return s

name = s1(x) + '-' + s2(x, y) + '-' + s3()
print name
```

ΘΕΜΑ Δ_1_2015 (Υπολογισμός κοινοχρήστων πολυκατοικίας)

Μια πολυκατοικία έχει 5 ορόφους, με 8 διαμερίσματα (Δ1, Δ2, ...,Δ8) σε κάθε όροφο. Τα διαμερίσματα Δ1 όλων των ορόφων έχουν το ίδιο εμβαδό (Ε1), τα διαμερίσματα Δ2 όλων των ορόφων έχουν το ίδιο εμβαδό (Ε2) κ.ο.κ. Το ποσό των κοινοχρήστων της πολυκατοικίας κατανέμεται στους 5 ορόφους, σύμφωνα με το ποσοστό συμμετοχής του κάθε ορόφου, όπως φαίνεται στον Πίνακα ΙΙΙ.

Όροφος	Ποσοστό συμμετοχής
1ος	5%
2ος	15%
3ος	20%
4ος	25%
5ος	35%

Πίνακας ΙΙΙ

Το ποσό των κοινοχρήστων του κάθε ορόφου κατανέμεται στα διαμερίσματα του ορόφου αυτού, ανάλογα με το εμβαδό του καθενός διαμερίσματος.

Να γράψετε πρόγραμμα, το οποίο:

Δ1. Να περιλαμβάνει κατάλληλο τμήμα δηλώσεων.

Μονάδες 2

Δ2. Να ζητάει:

α. Το συνολικό ποσό κοινοχρήστων της πολυκατοικίας (μονάδα 1).

β. Τα εμβαδά Ε1, Ε2, ... Ε8. (μονάδα 1).

Μονάδες 2

Δ3. Να υπολογίζει το ποσό των κοινοχρήστων που αναλογεί σε κάθε όροφο της πολυκατοικίας.

Μονάδες 4

Δ4. Να υπολογίζει το ποσό των κοινοχρήστων που αναλογεί σε κάθε διαμέρισμα της πολυκατοικίας.

Μονάδες 7

Δ5. Να αναζητά και να εμφανίζει τον αριθμό ορόφου (1-5) και τον αριθμό διαμερίσματος (1-8) ενός διαμερίσματος στο οποίο αναλογεί ποσό κοινοχρήστων μεγαλύτερο του μέσου όρου όλης της πολυκατοικίας. Η αναζήτηση να ξεκινά από τον 1ο όροφο και για κάθε όροφο να ξεκινά από το διαμέρισμα Δ8. Η αναζήτηση να τερματίζεται μόλις βρεθεί ένα τέτοιο διαμέρισμα.

Μονάδες 5

```
# ----------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ----------------------------------------------------------------------
# Όνομα:        2015_1_D
# Περιγραφή:    Υπολογισμός κοινοχρήστων πολυκατοικίας
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ----------------------------------------------------------------------

# Δ1,2
total = input ("Δώσε τα κοινόχρηστα της πολυκατοικίας = ")
surface = []
for i in range(8):
    surface.append(input("Δώσε το εμβαδόν των διαμερισμάτων = "))
# Δ3
floors = []
ratio = [0.05, 0.15, 0.2, 0.25, 0.35]
for floor in range(5):
    floors.append(ratio[floor] * total)
# Δ4
floor_surface = 0
for num in range(8):
    floor_surface += surface[num]
apartments = []
for floor in range(5):
    apartments.append([])
    for num in range(8):
        apartments[floor].append(floors[floor] * surface[num] / floor_surface)
# Δ5
xsum = 0
for floor in range(5):
    for num in range(8):
        xsum += apartments[floor][num]
average = xsum * 1.0 / 40
found = False
floor = 0
while floor < 5 and not found:
    num = 8
    while num >= 0 and not found:
        if apartments[floor][num] > average:
            print floor, num
            found = True
        num -= 1
    floor += 1

# Εκτύπωση κοινοχρήστων αν και δε ζητείται
for i in range(5):
    for j in range(8):
        print round(apartments[i][j],2),
    print
```

ΘΕΜΑ Γ_0_2016_ΝΕΟ (Απόθεμα & Πωλήσεις Η/Υ)

Μία εταιρεία πληροφορικής προσφέρει υπολογιστές σε τιμές οι οποίες μειώνονται ανάλογα με την ποσότητα της παραγγελίας, όπως φαίνεται στον παρακάτω πίνακα:

ΠΟΣΟΤΗΤΑ	ΤΙΜΗ ΜΟΝΑΔΑΣ
1-50	580
51-100	520
101-200	470
Πάνω από 200	440

Να κατασκευάσετε πρόγραμμα το οποίο:

Γ1. Να περιλαμβάνει κατάλληλο τμήμα δηλώσεων.

Μονάδες 2

Γ2. Να διαβάζει τον αριθμό υπολογιστών που έχει προς πώληση (απόθεμα), ελέγχοντας ότι δίνεται θετικός αριθμός

Μονάδες 2

Γ3. Για κάθε παραγγελία, να διαβάζει την απαιτούμενη ποσότητα και, εφόσον το απόθεμα επαρκεί για την κάλυψη της ποσότητας να εκτελεί την παραγγελία με την ποσότητα που ζητήθηκε. Αν το απόθεμα δεν επαρκεί, διατίθεται στον πελάτη το διαθέσιμο απόθεμα. Η εισαγωγή παραγγελιών τερματίζεται, όταν εξαντληθεί το απόθεμα.

Μονάδες 6

Για κάθε παραγγελία να εμφανίζει:

Γ4. το κόστος της παραγγελίας Μονάδες 4

Γ5. το επιπλέον ποσό που θα κόστιζε η παραγγελία, εάν ο υπολογισμός γινόταν κλιμακωτά με τις τιμές που φαίνονται στον πίνακα.

Μονάδες 6

```
# ----------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ----------------------------------------------------------------------
# Όνομα:        2016_0_C_NEW
# Περιγραφή:    Απόθεμα & Πωλήσεις Η/Υ
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ----------------------------------------------------------------------

# Γ1,2
stock = input("Δώσε το απόθεμα των ΗΥ = ")
while stock < 0:
    print "Λάθος δεδομένα "
    stock = input("Δώσε το απόθεμα των ΗΥ = ")

# Γ3
def cost(q):
    if q <= 50:
        return q * 580
    elif q <= 100:
        return q * 520
    elif q <= 200:
        return q * 470
    else:
        return q * 440


def scale_cost(q):
    if q <= 50:
        return q * 580
    elif q <= 100:
        return 50*580 + (q - 50)*520
    elif q <= 200:
        return 50*580 + 100*520 + (q - 100) * 470
    else:
        return 50*580 + 100*520 + 200*470 + (q-200) * 440

while stock > 0:
    order = input("Δώσε τεμάχια παραγγελίας = ")

    if order <= stock:
        print "Παραγγελία εκτελέστηκε Πλήρως"
        # Γ4
        print "Κόστος Παραγγελίας", "Διαφορά από κλιμακωτή χρέωση"
        print cost(order), scale_cost(order) - cost(order)
        stock -= order
    else:
        print stock, " τεμάχια διαθέσιμα μόνο"
        # Γ4
        print "Κόστος Παραγγελίας", "Διαφορά από κλιμακωτή χρέωση"
        print cost(stock), scale_cost(stock) - cost(stock)
        stock = 0
```

ΘΕΜΑ Δ_0_2016_ΝΕΟ (Στατιστικά στοιχεία πρόσβασης ΠΣΔ)

Το Πανελλήνιο Σχολικό Δίκτυο παρέχει πρόσβαση στο Διαδίκτυο (Ίντερνετ) σε 150.000 μαθητές και διατηρεί τα στοιχεία τους, καθώς και στατιστικά στοιχεία, σχετικά με την πρόσβασή τους στο Διαδίκτυο.

Να κατασκευάσετε πρόγραμμα το οποίο:

Δ1. Να περιλαμβάνει κατάλληλο τμήμα δηλώσεων.

Μονάδες 2

Δ2. Για κάθε μαθητή να διαβάζει:

α) τον αλφαριθμητικό κωδικό του και να τον καταχωρίζει σε μονοδιάστατο πίνακα με όνομα ΚΩΔ

β) το φύλο του, «Α» αν είναι αγόρι και «Κ» αν είναι κορίτσι, και να το καταχωρίζει σε μονοδιάστατο πίνακα με όνομα Φ

γ) τον συνολικό χρόνο πρόσβασής του στο Διαδίκτυο ανά μήνα, για ένα έτος, και να τον καταχωρίζει σε δισδιάστατο πίνακα ΧΡ.

Μονάδες 3

Δ3. Να υπολογίζει και να καταχωρίζει σε πίνακα ΣΧ το συνολικό ετήσιο χρόνο πρόσβασης κάθε μαθητή.

Μονάδες 3

Δ4. Να εμφανίζει τον κωδικό του αγοριού με το μεγαλύτερο συνολικό χρόνοπρόσβασης και, στη συνέχεια, τον κωδικό του κοριτσιού με το μεγαλύτερο συνολικό χρόνο πρόσβασης, καλώντας τη συνάρτηση ΘΕΣΗ_ΜΑΧ, που περιγράφεται στο ερώτημα Δ5, μία φορά για τα αγόρια και μία για τα κορίτσια.

Μονάδες 4

Δ5. Να αναπτύξετε συνάρτηση ΘΕΣΗ_ΜΑΧ η οποία:

α) να δέχεται ως παραμέτρους: τον πίνακα του φύλου, τον πίνακα του συνολικού ετήσιου χρόνου πρόσβασης των μαθητών και τον χαρακτήρα «Α» ή «Κ» που αντιστοιχεί στο φύλο (μονάδες 2)

β) να βρίσκει τη θέση της μέγιστης τιμής του ετήσιου χρόνου πρόσβασης αγοριών ή κοριτσιών, ανάλογα με την τιμή «Α» ή «Κ» του φύλου

(μονάδες 4)

γ) να επιστρέφει τη θέση της μέγιστης τιμής (μονάδες 2)

Μονάδες 8

(Σημείωση: Δεν απαιτείται έλεγχος εγκυρότητας. Να θεωρήσετε ότι όλες οι εισαγωγές γίνονται σωστά και όλες οι συνολικές τιμές χρόνου πρόσβασης είναι μοναδικές).

```
# ----------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ----------------------------------------------------------------------
# Όνομα:        2016_0_D_NEW
# Περιγραφή:    Στατιστικά στοιχεία πρόσβασης ΠΣΔ
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ----------------------------------------------------------------------

# Δ1
# Δ2,α
code = []
for i in range(150000):
    code.append(raw_input("Δώσε τον κωδικό του μαθητή = "))

# Δ2,β
sex = []
for i in range(150000):
    sex.append(raw_input("Δώσε το φύλο Α/Κ = "))

# Δ2,γ
time = []
for s in range(150000):
    time.append([])
    for m in range(12):
        time[s].append(input("Δώσε το χρόνο πρόσβασης = "))

# Δ3
total_time = []
for s in range(150000):
    tsum = 0
    for m in range(12):
        tsum += time[s][m]
    total_time.append(tsum)


# Δ5
def max_pos(sex, time_list, gender):
    tmax = time_list[0]
    pos = 0
    for s in range(1, 150000):
        if gender == sex[s]:
            if tmax < time_list[s]:
                tmax = time_list[s]
                pos = s
    return pos

# Δ4
print code[max_pos(sex, total_time, 'A')]
print code[max_pos(sex, total_time, 'K')]
```

ΘΕΜΑ Γ_0_2016_ΠΑΛΑΙΟ (Διαθέσιμη χωρητικότητα σκληρού δίσκου)

Ένας μαθητής αγόρασε έναν εξωτερικό δίσκο χωρητικότητας 1000 GB, προκειμένου να αποθηκεύσει σε αυτόν ψηφιακά αρχεία.

Να γραφεί πρόγραμμα σε ΓΛΩΣΣΑ, το οποίο:

Γ1. α. Να περιλαμβάνει κατάλληλο τμήμα δηλώσεων.

(μονάδες 2)

β. Για κάθε ψηφιακό αρχείο που θέλει να αποθηκεύσει ο μαθητής στον εξωτερικό δίσκο, να διαβάζει το όνομά του και το μέγεθός του (σε GB) και να ελέγχει, αν επαρκεί η διαθέσιμη χωρητικότητα του εξωτερικού δίσκου. Εφόσον επαρκεί, να εμφανίζει το μήνυμα «Επιτρεπτή αποθήκευση» και να υπολογίζει τη νέα διαθέσιμη χωρητικότητα του εξωτερικού δίσκου. Να τερματίζει τον έλεγχο της αποθήκευσης ψηφιακών αρχείων στον εξωτερικό δίσκο, όταν το μέγεθος του αρχείου που θέλει να αποθηκεύσει ο μαθητής είναι μεγαλύτερο από τη διαθέσιμη χωρητικότητα του εξωτερικού δίσκου.

(μονάδες 6)

Μονάδες 8

Γ2. Να υπολογίζει και να εμφανίζει το ποσοστό του αριθμού των αρχείων που αποθηκεύτηκαν και έχουν μέγεθος μεγαλύτερο των 10 GB.

Μονάδες 4

Γ3. Να βρίσκει και να εμφανίζει τα ονόματα των δύο μικρότερων σε μέγεθος αρχείων που αποθηκεύτηκαν στον εξωτερικό δίσκο.

Μονάδες 8

Να θεωρήσετε ότι:

α) θα αποθηκευτούν τουλάχιστον δύο αρχεία στον εξωτερικό δίσκο,

β) τα μεγέθη όλων των αρχείων που αποθηκεύονται, είναι διαφορετικά μεταξύ τους.

```
# ------------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ------------------------------------------------------------------------
# Όνομα:        2016_0_C_OLD
# Περιγραφή:    Διαθέσιμη χωρητικότητα σκληρού δίσκου
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ------------------------------------------------------------------------

# Γ1
total = 1000
file_name = raw_input("Δώσε το όνομα του αρχείου = ")
file_size = input("Δώσε το μεγεθος του αρχείου = ")
files = 0
count = 0
min1_name = file_name
min1_size = file_size

while file_size <= total:

    print "Επιτρεπτή αποθήκευση"
    files += 1
    if file_size > 10:
        count += 1

    total -= file_size
    file_name = raw_input("Δώσε το όνομα του αρχείου = ")
    file_size = input("Δώσε το μεγεθος του αρχείου = ")

    # Αρχικοποίηση του min2
    if count == 1:
        min2_name = file_name
        min2_size = file_size

    if file_size < min1_size:
        min2_size = min1_size
        min2_nam2 = min1_name
        min1_size = file_size
        min1_name = file_name

print total, count
print min1_name, min1_size
print min2_name, min2_size
```

ΘΕΜΑ Δ_0_2016_ΠΑΛΑΙΟ (Εθελοντές Περιβαλλοντικής Οργάνωσης)

Μια περιβαλλοντική οργάνωση έχει εκπαιδεύσει δέκα (10) εθελοντές οι οποίοι θα ενημερώσουν το κοινό σε θέματα που αφορούν την προστασία του περιβάλλοντος.

Να γράψετε πρόγραμμα σε ΓΛΩΣΣΑ, το οποίο:

Δ1. α. Να περιλαμβάνει κατάλληλο τμήμα δηλώσεων.

(μονάδα 1)

β. Για κάθε εθελοντή, να διαβάζει το όνομά του και τον αριθμό των ατόμων που ενημέρωσε κάθε μήνα, στη διάρκεια του προηγούμενου έτους (δεν απαιτείται έλεγχος εγκυρότητας).

(μονάδες 2)

Μονάδες 3

Δ2. Για κάθε μήνα, να εμφανίζει το συνολικό αριθμό ατόμων που ενημέρωσαν οι δέκα (10) εθελοντές. Ο υπολογισμός του συνολικού αριθμού ατόμων, που ενημέρωσαν κάθε μήνα, να γίνει με κλήση κατάλληλης συνάρτησης.

Μονάδες 3

Δ3. Να εμφανίζει τα ονόματα των τριών εθελοντών που ενημέρωσαν τα περισσότερα άτομα, κατά τη διάρκεια του προηγούμενου έτους. Να θεωρήσετε ότι κάθε εθελοντής ενημέρωσε διαφορετικό συνολικό αριθμό ατόμων κατά τη διάρκεια του έτους.

Μονάδες 9

Δ4. Να κατασκευάσετε τη συνάρτηση του ερωτήματος Δ2.

Μονάδες 5

Να θεωρήσετε ότι κάθε άτομο ενημερώνεται μόνο από ένα εθελοντή.

```
# ---------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ---------------------------------------------------------------------
# Όνομα:        2016_0_D_OLD
# Περιγραφή:    Εθελοντές Περιβαλλοντικής Οργάνωσης
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ---------------------------------------------------------------------

# Δ1 α,β
info = []
names = []
for i in range(10):
    info.append([])
    for j in range(12):
        info[i].append(input("Δώσε τον αριθμό ατόμων που ενημερώθηκαν = "))


# Δ4
def total(a_list, col):
    xsum = 0
    for i in range(10):
        xsum += a_list[i][col]
    return xsum

# Δ2
for j in range(12):
    print j, total(info, j)

# Δ3
year = []
for i in range(10):
    xsum = 0
    for j in range(12):
        xsum += info[i][j]
    year.append(xsum)

imax3 = []
for k in range(3):
    imax = 0
    xmax = 0
    for i in range(10):
        if year[i] > xmax and i not in imax3:
            xmax = year[i]
            ixmax = i
    imax3.append(imax)
for i in range(3):
    print names[imax3[i]]
```

ΘΕΜΑ Γ_1_2016_ΝΕΟ (Μελέτη δείγματος ναυτικών λέξεων)

Στο πλαίσιο μιας μελέτης, ένας φιλόλογος θέλει να ελέγξει τη χρήση ενός δείγματος εκατό (100) ναυτικών λέξεων σε σύγχρονα νεοελληνικά κείμενα. Για τον σκοπό αυτό:

Γ1. Να κατασκευάσετε υποπρόγραμμα, με όνομα ΑΝΑΖΗΤΗΣΗ, το οποίο να δέχεται

ένα μονοδιάστατο πίνακα χαρακτήρων Π[100],

μια ακέραια μεταβλητή Ν,

μια αλφαριθμητική μεταβλητή Χ

και να επιστρέφει

μια λογική μεταβλητή ΒΡΕΘΗΚΕ και

μια ακέραια μεταβλητή ΘΕΣΗ.

Το υποπρόγραμμα να αναζητά μια λέξη, την τιμή της μεταβλητής Χ στις θέσεις 1 έως Ν του πίνακα Π. Αν βρεθεί η λέξη, το υποπρόγραμμα να επιστρέφει την τιμή ΑΛΗΘΗΣ και τη θέση που βρέθηκε. Αν δεν βρεθεί, να επιστρέφει την τιμή ΨΕΥΔΗΣ και την τιμή 0.

Μονάδες 5

Στη συνέχεια να κατασκευάσετε κύριο πρόγραμμα το οποίο :

Γ2. Να ζητά 100 ναυτικές λέξεις και να τις καταχωρίζει σε πίνακα ΛΕΞΕΙΣ[100]. Κάθε λέξη που δίνεται να τη δέχεται, μόνο εφόσον ελέγξει ότι δεν έχει ήδη καταχωριστεί στον πίνακα. Ο έλεγχος να γίνεται με τη χρήση του υποπρογράμματος ΑΝΑΖΗΤΗΣΗ.

Μονάδες 5

Γ3. Να ζητά, με τη σειρά, τις λέξεις ενός νεοελληνικού κειμένου. Η εισαγωγή να τερματίζεται όταν δοθεί ως λέξη η ακολουθία χαρακτήρων «ΤΕΛΟΣ_ΚΕΙΜΕΝΟΥ».

Μονάδες 2

Γ4. Να εμφανίζει τις σπανιότερες ναυτικές λέξεις του δείγματος που υπάρχουν στο νεοελληνικό κείμενο, δηλαδή τις λέξεις με τη μικρότερη συχνότητα εμφάνισης, χρησιμοποιώντας κατάλληλα το υποπρόγραμμα ΑΝΑΖΗΤΗΣΗ.

Μονάδες 8

```
# ----------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ----------------------------------------------------------------------
# Όνομα:        2016_1_C_NEW
# Περιγραφή:    Μελέτη δείγματος ναυτικών λέξεων
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ----------------------------------------------------------------------

# Γ1
def search(a_list, n , x):
    # Η μεταβλητή pos είναι αρκετή για την αναζήτηση
    pos = -1
    i = 0
    if n <= len(a_list):
        while i <= n and pos == -1:
            if a_list[i] == x:
```

```
                pos = i
            else:
                i += 1
    return pos

# Γ2
# Η Python διαθέτει τον τελεστή in
##naval_words = []
##for w in range(100):
##    word = raw_input("Δώσε μία ναυτική λέξη = ")
##    while word in naval_words:
##        print "Η λέξη αυτή υπάρχει δώσε μία άλλη λέξη"
##        word = raw_input("Δώσε μία ναυτική λέξη = ")
##    naval_words.append(word)

# Χωρίς χρήση του τελεστή in
naval_words = []
word = raw_input("Δώσε μία ναυτική λέξη = ")
naval_words.append(word)
for w in range(100):
    word = raw_input("Δώσε μία ναυτική λέξη = ")
    while search(naval_words, w, word) != -1:
        print "Η λέξη αυτή υπάρχει δώσε μία άλλη λέξη"
        word = raw_input("Δώσε μία ναυτική λέξη = ")
    naval_words.append(word)

# Γ3
words = []
word = raw_input("Δώσε μία λέξη text_end για τέλος = ")
while word != 'text_end':
    words.append(word)
    word = raw_input("Δώσε μία λέξη text_end για τέλος = ")

# Γ4
f = []  # Συχνότητα εμφάνισης
for nw in naval_words:
    count = 0
    for word in words:
        if nw == word:
            count += 1
    f.append(count)

wmin = len(words) + 1
count = 0
for i in range(len(f)):
    if f[i] < wmin:
        wmin = f[i]
        count += 1
if count == 0:
    print "Δε βρέθηκε καμία ναυτική λεξη στο κείμενο"
else:
    print "Οι σπανιότερες ναυτικές λέξεις που βρέθηκαν είναι"
    for i in range(len(f)):
        if f[i] == wmin:
            print naval_words[i],
```

ΘΕΜΑ Δ_1_2016_ΝΕΟ (Ταξινόμηση κεραμικών σε αρχαιολογικό χώρο Πύλου)

Στον αρχαιολογικό χώρο της Πύλου διασώθηκαν θραύσματα κεραμικών πινακίδων στα οποία είχαν καταγραφεί σε γραμμές βασικά αγαθά με τις ποσότητες τους, τα οποία είχε συλλέξει η πόλη κατά τη διάρκεια καλλιεργητικών περιόδων. Σε κάθε θραύσμα, αναφέρονται τα πλήρη στοιχεία (όνομα αγαθού, περίοδος, ποσότητα) για ένα ή περισσότερα αγαθά. Βρέθηκαν στοιχεία για δεκαπέντε (15) βασικά αγαθά και πέντε (5) καλλιεργητικές περιόδους. Όλα τα αγαθά υπάρχουν και στις πέντε περιόδους

Σε κάθε γραμμή οι πρώτοι δέκα χαρακτήρες αντιστοιχούν στο όνομα του αγαθού, ο ενδέκατος στην καλλιεργητική περίοδο και ο δωδέκατος στην ποσότητα που συλλέχτηκε. Οι πέντε καλλιεργητικές περίοδοι αναπαρίστανται από τους χαρακτήρες Α, Β, Γ, Δ και Ε. Η ποσότητα που συλλέχτηκε αναπαρίσταται από τους χαρακτήρες Ι, Κ, Λ, Μ, Ν, Ξ και Ο. Έχει βρεθεί ότι η ποσότητα που αντιστοιχεί σε αυτούς είναι: Ι = 10, Κ = 50, Λ = 100, Μ = 500, Ν = 1.000, Ξ = 5.000 και Ο = 10.000.

Συνολικά τα στοιχεία των θραυσμάτων μπορούν να αναπαρασταθούν με ένα δισδιάστατο πίνακα Π[75,12]. Κάθε γραμμή του πίνακα περιέχει τα στοιχεία των αγαθών (όνομα αγαθού, καλλιεργητική περίοδος, ποσότητα). Κάθε στοιχείο του πίνακα περιέχει ένα μόνο χαρακτήρα.

Να γράψετε πρόγραμμα σε ΓΛΩΣΣΑ το οποίο:

Δ1. α. Να περιλαμβάνει κατάλληλο τμήμα δηλώσεων.

(μονάδα 1)

β. Να εισάγει σε πίνακα χαρακτήρων Π[75,12] τα στοιχεία των αγαθών που βρέθηκαν στα θραύσματα των πινακίδων.

(μονάδες 2)

Μονάδες 3

Δ2. Να ταξινομεί κατά αύξουσα σειρά τον πίνακα Π, με βάση την καλλιεργητική περίοδο, και, για την ίδια καλλιεργητική περίοδο, να ταξινομεί τα αγαθά, με βάση τον πρώτο χαρακτήρα κάθε αγαθού. (Θεωρήστε ότι ο πρώτος χαρακτήρας κάθε αγαθού είναι μοναδικός).

Μονάδες 6

Δ3. α. Να δημιουργεί έναν πίνακα ακεραίων Α[75]. Κάθε στοιχείο του πίνακα Α αντιστοιχεί σε μια γραμμή του ταξινομημένου πίνακα Π και περιέχει την αντίστοιχη ποσότητα του αγαθού που συλλέχτηκε. Η μετατροπή της ποσότητας από χαρακτήρα σε αριθμό να γίνει με βάση την αντιστοιχία που δόθηκε παραπάνω.

(μονάδες 2)

β. Να βρίσκει και να εμφανίζει για κάθε αγαθό το πρώτο γράμμα του ονόματός του και την καλλιεργητική του περίοδο με τη μέγιστη ποσότητα που συλλέχτηκε. (Θεωρήστε ότι η μέγιστη ποσότητα κάθε αγαθού είναι μοναδική).

(μονάδες 4)

Μονάδες 6

Δ4. Να δημιουργεί έναν πίνακα ακεραίων Σ[15]. Κάθε στοιχείο του πίνακα Σ αντιστοιχεί σε ένα αγαθό (όπως αυτό εμφανίζεται στις δεκαπέντε πρώτες σειρές του πίνακα Π) και περιέχει την συνολική ποσότητα του αγαθού που συλλέχτηκε στις πέντε καλλιεργητικές περιόδους.

Μονάδες 5

```
# ----------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ----------------------------------------------------------------------
# Όνομα:        2016_1_D_NEW
# Περιγραφή:    Ταξινόμηση κεραμικών σε αρχαιολογικό χώρο Πύλου
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ----------------------------------------------------------------------

# Δ1 α,β
p = []
for i in range(75):
    p.append([])
    for j in range(12):
        p[i].append(raw_input("Δώσε τα στοιχεία των πινακίδων = "))

# Δ2
for i in range(74):
    for j in range(74, i, -1):
        if (p[j][10] < p[j-1][10] or
           (p[j][10] == p[j-1][10] and p[j][10] < p[j-1][10])):
            for k in range(12):
                p[j-1][k], p[j][k] = p[j][k], p[j-1][k]

# Δ3,α
a = []
for i in range(75):
    c = p[i][11]
    if c == 'I':
        a[i] = 10
    elif c == 'K':
        a[i] = 50
    elif c == 'L':
        a[i] = 100
    elif c == 'M':
        a[i] = 500
    elif c == 'N':
        a[i] = 1000
    elif c == 'J':
        a[i] = 5000
    else:
        a[i] = 10000

# Δ3,β
for j in range(15):
    xmax = a[j]
    pmax = p[j][10]
    for i in range(0, 75, 15):
        if a[i] > xmax:
            xmax = a[i]
            pmax = p[i][10]
    print p[j][1], pmax, xmax
```

```
# Δ4
s = []
for j in range(15):
    xsum = 0
    for i in range(0, 75, 15):
        xsum += a[i]
    s.append(xsum)
```

ΘΕΜΑ Γ_1_2016_ΠΑΛΑΙΟ (Ενοικίαση δωματίων ξενοδοχείου)

Ένα ξενοδοχείο χρεώνει την ενοικίαση των δωματίων του ανάλογα με τον αριθμό των ημερών ενοικίασης και την τουριστική περίοδο, σύμφωνα με τον παρακάτω πίνακα:

ΑΡΙΘΜΟΣ ΗΜΕΡΩΝ	ΤΟΥΡΙΣΤΙΚΗ ΠΕΡΙΟΔΟΣ	
	ΧΑΜΗΛΗ	ΥΨΗΛΗ
1-3	40€ ανά ημέρα	70€ ανά ημέρα
4-7	30€ ανά ημέρα	55€ ανά ημέρα
>7	25€ ανά ημέρα	50€ ανά ημέρα

Να αναπτύξετε πρόγραμμα σε ΓΛΩΣΣΑ το οποίο:

Γ1. Να περιλαμβάνει κατάλληλο τμήμα δηλώσεων.

Μονάδες 2

Γ2. Για καθεμιά από τις 500 κρατήσεις του ξενοδοχείου κατά το προηγούμενο έτος:

α. Να διαβάζει τον αριθμό των ημερών ενοικίασης καθώς και την τουριστική περίοδο που έγινε η κράτηση, εξασφαλίζοντας ότι η επιτρεπτή τιμή για την τουριστική περίοδο είναι ΧΑΜΗΛΗ ή ΥΨΗΛΗ.

(μονάδες 3)

β. Να καλεί υποπρόγραμμα με είσοδο τον αριθμό των ημερών ενοικίασης και την τουριστική περίοδο, το οποίο να υπολογίζει, με βάση τον προηγούμενο πίνακα, τη χρέωση της κράτησης. Ο υπολογισμός της χρέωσης δεν γίνεται κλιμακωτά.

(μονάδες 2)

γ. Να εμφανίζει τη χρέωση της κράτησης.

(μονάδα 1)

Μονάδες 6

Γ3. Να υπολογίζει και να εμφανίζει τη συνολική χρέωση των κρατήσεων του ξενοδοχείου για καθεμιά τουριστική περίοδο του προηγούμενου έτους.

Μονάδες 4

Γ4. Να κατασκευάσετε το υποπρόγραμμα του ερωτήματος Γ2.β.

Μονάδες 8

```
# ----------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ----------------------------------------------------------------------
# Όνομα:        2016_1_C_OLD
# Περιγραφή:    Ενοικίαση δωματίων ξενοδοχείου
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ----------------------------------------------------------------------

# Γ1
total_low = 0
total_high = 0

# Γ4
def check_out(days, period):
    if period in ['L', 'LOW']:
        if days <= 3:
            check = 40 * days
        elif days <= 7:
            check = 30 * days
        elif days > 7:
            check = 25 * days
    elif period in ['H', 'HIGH']:
        if days <= 3:
            check = 70 * days
        elif days <= 7:
            check = 55 * days
        elif days > 7:
            check = 50 * days
    return check

# Γ2,α
for i in range(500):
    days = input('Δώσε ημέρες ενοικίασης = ')
    period = raw_input('Δώσε περίοδο ενοικίασης Low(L),High(H) = ')
    while period.upper() not in ['L', 'H', 'LOW', 'HIGH']:
        print 'Λάθος περίοδος'
        period = raw_input('Δώσε περίοδο ενοικίασης Low(L),High(H) = ')

    # Γ2,β
    period = period.upper()
    check = check_out(days, period)

    # Γ2,γ
    print 'Θα πληρώσετε = ', check

    # Γ3
    if period in ['L', 'LOW']:
        total_low += check
    else:
        total_high += check
    print "Χαμηλή περίοδος συνολο = ", total_low
    print "Υψηλή περίοδος συνολο = ", total_high
```

ΘΕΜΑ Δ_1_2016_ΠΑΛΑΙΟ (Υποκαταστήματα - Πωλητές - Πωλήσεις)

Μια εταιρεία έχει δύο υποκαταστήματα, ένα στην Αθήνα και ένα στη Θεσσαλονίκη. Σε κάθε υποκατάστημα εργάζονται 10 πωλητές.

Να αναπτύξετε αλγόριθμο σε ψευδογλώσσα, ο οποίος:

Δ1. Για καθέναν από τους 20 πωλητές της εταιρείας, να διαβάζει το όνομά του και τον κωδικό του και να τα καταχωρίζει σε κατάλληλο δισδιάστατο πίνακα, έτσι ώστε στις πρώτες 10 γραμμές του πίνακα να υπάρχουν τα στοιχεία των πωλητών του υποκαταστήματος της Αθήνας και στις επόμενες 10 τα στοιχεία των πωλητών της Θεσσαλονίκης. Να θεωρήσετε ότι όλα τα ονόματα και όλοι οι κωδικοί είναι διαφορετικοί μεταξύ τους.

Μονάδες 2

Δ2. Για κάθε παραγγελία της εταιρείας στη διάρκεια του προηγούμενου έτους, να διαβάζει τον κωδικό του πωλητή. Αν ο κωδικός ανήκει σε πωλητή της εταιρείας, να διαβάζει το ποσό της αντίστοιχης παραγγελίας που πήρε ο πωλητής (δεν απαιτείται έλεγχος εγκυρότητας) ή, διαφορετικά, να εμφανίζει το μήνυμα «Άγνωστος κωδικός». Η επαναληπτική διαδικασία να τερματίζεται όταν δοθεί, ως κωδικός πωλητή, η τιμή ΤΕΛΟΣ.

Μονάδες 8

Δ3. Να υπολογίζει τις συνολικές πωλήσεις κάθε πωλητή στη διάρκεια του προηγούμενου έτους και να τις εμφανίζει μαζί με το όνομά του. Να θεωρήσετε ότι κάθε πωλητής πήρε παραπάνω από μία παραγγελία στη διάρκεια του προηγούμενου έτους.

Μονάδες 4

Δ4. Για κάθε υποκατάστημα να βρίσκει και να εμφανίζει τα ονόματα των τριών πωλητών με τις μεγαλύτερες συνολικές πωλήσεις στη διάρκεια του προηγούμενου έτους. Να θεωρήσετε ότι οι συνολικές πωλήσεις όλων των πωλητών είναι διαφορετικές μεταξύ τους.

Μονάδες 6

```
# ---------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ---------------------------------------------------------------------
# Όνομα:        2016_1_D_OLD
# Περιγραφή:    Υποκαταστήματα - Πωλητές - Πωλήσεις
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ---------------------------------------------------------------------

# Δ1
sellers = []
flag = False

print 'Δώσε τα στοιχεία των πωλητών της Αθήνας'
for i in range(20):
    sellers.append([])
    if i >= 10 and not flag:
        print 'Δώσε τα στοιχεία των πωλητών της Θεσ/νίκης'
        flag = True
    sellers[i].append(raw_input('Δώσε το όνομα του πωλητή = '))
    sellers[i].append(raw_input('Δώσε τον κωδικό του πωλητή = '))

# test data comment out
sellers = [
    ['a', '1'], ['b', '2'], ['c', '3'], ['d', '4'], ['e', '5'],
    ['f', '6'], ['g', '7'], ['h', '8'], ['j', '9'], ['k', '10'],
    ['aa', '11'], ['bb', '22'], ['cc', '33'], ['dd', '44'],
    ['ee', '55'], ['ff', '66'], ['gg', '77'], ['hh', '88'],
    ['jj', '99'], ['kk', '1010']]


# Δ2
def find_code(sellers, key):
    pos = -1
    i = 0
    while i < 20 and pos == -1:
        if sellers[i][1] == key:
            pos = i
        else:
            i += 1
    return pos

# Δ3
sales = [0] * 20
code = raw_input('Δώσε κωδικό πωλητή ή (END) για τέλος = ')

while code.upper() != 'END':
    r = find_code(sellers, code)
    if r != -1:
        order = input('Δώσε ποσό παραγγελίας = ')
        sales[r] += order
    code = raw_input('Δώσε κωδικό πωλητή = ')
```

```
for i in range(20):
    print sellers[i][0], sales[i]

# Δ4
for i in range(9):
    for j in range(9, i, -1):
        print i, j
        if sales[j-1] < sales[j]:
            sales[j], sales[j-1] = sales[j-1], sales[j]
            sellers[j][0], sellers[j-1][0] = sellers[j-1][0], sellers[j][0]
            sellers[j][1], sellers[j-1][1] = sellers[j-1][1], sellers[j][1]

for i in range(10,19):
    for j in range(19, i, -1):
        if sales[j-1] < sales[j]:
            sales[j], sales[j-1] = sales[j-1], sales[j]
            sellers[j][0], sellers[j-1][0] = sellers[j-1][0], sellers[j][0]
            sellers[j][1], sellers[j-1][1] = sellers[j-1][1], sellers[j][1]

print 'Οι 3 πρώτοι στην Αθήνα'
for i in range(3):
    print i, sellers[i][0], sales[i]


print 'Οι 3 πρώτοι στη Θεσ/νίκη'
for i in range(10,13):
    print sellers[i][0], sales[i]
```

ΘΕΜΑ Γ_0_2017 (Αποτελέσματα σχολικού πρωταθλήματος Βόλεϊ)

Στο πλαίσιο ενός τοπικού σχολικού πρωταθλήματος βόλεϊ συμμετέχουν 5 σχολεία, αριθμημένα από το 1 έως το 5. Κάθε σχολείο παίζει μία φορά με όλα τα υπόλοιπα. Άρα θα πραγματοποιηθούν συνολικά 10 αγώνες. Νικητής ενός αγώνα είναι το σχολείο που έχει κερδίσει 3 σετ. Ο νικητής παίρνει 2 βαθμούς και ο ηττημένος 1 βαθμό. Κάθε αγώνας προσδιορίζεται από τα σχολεία που παίζουν μεταξύ τους και το αποτέλεσμα του αγώνα σε σετ. Για παράδειγμα, η σειρά των στοιχείων: 4, 5, 1, 3 σημαίνει ότι το σχολείο 4 έπαιξε με το σχολείο 5 και έχασε τον αγώνα με 1 σετ υπέρ και 3 κατά. Αυτό αντίστοιχα σημαίνει ότι το σχολείο 5 κέρδισε τον αγώνα με το σχολείο 4 με 3 σετ υπέρ και 1 σετ κατά. Τα δεδομένα των αγώνων αποθηκεύονται σε έναν δισδιάστατο πίνακα Α[5,3], όπου κάθε γραμμή αντιστοιχεί σε ένα σχολείο. Η τελική μορφή του πίνακα Α θα περιέχει για κάθε σχολείο, στην πρώτη (1η) στήλη τη βαθμολογία του (το άθροισμα των βαθμών του), στη δεύτερη (2η) το άθροισμα των σετ υπέρ και στην τρίτη (3η) το άθροισμα των σετ κατά, από όλους τους αγώνες.

Να κατασκευάσετε πρόγραμμα σε ΓΛΩΣΣΑ το οποίο:

Γ1. α) Να περιλαμβάνει κατάλληλο τμήμα δηλώσεων. (μονάδες 2)

β) Να διαβάζει τα ονόματα των 5 σχολείων και να τα καταχωρίζει στον πίνακα ΟΝ [5]. Η σειρά των σχολείων καθορίζει την αρίθμησή τους (1έως 5). (μονάδες 2)

γ) Να αρχικοποιεί τον πίνακα Α[5,3]. (μονάδες 2)

Μονάδες 6

Γ2. Να διαβάζει για κάθε αγώνα τη σειρά των 4 στοιχείων που τον προσδιορίζουν και να ενημερώνει τον πίνακα Α και για τα δύο σχολεία όπως περιγράφεται παραπάνω.

Μονάδες 6

Γ3. Να κατατάσσει τα σχολεία σε φθίνουσα σειρά ανάλογα με τη βαθμολογία τους και σε περίπτωση ισοβαθμίας να προηγείται το σχολείο με ταπερισσότερα σετ υπέρ.

Μονάδες 6

Γ4. Να εμφανίζει τα ονόματα των σχολείων, τη βαθμολογία τους, το άθροισμα των σετ υπέρ και το άθροισμα των σετ κατά, με βάση τη σειρά κατάταξής τους.

Μονάδες 2

Σημείωση: Θεωρείστε ότι δεν υπάρχει περίπτωση δύο σχολεία να έχουν και την ίδια βαθμολογία και τον ίδιο αριθμό σετ υπέρ.

```
# ----------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ----------------------------------------------------------------------
# Όνομα:        2017_0_C
# Περιγραφή:    Αποτελέσματα σχολικού πρωταθλήματος Βόλεϊ
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ----------------------------------------------------------------------

# Γ1 β,γ
names = []
a = []
for i in range(5):
    names.append(raw_input('Δώσε όνομα σχολείου = '))
    a.append([])
    for j in range(3):
        a[i].append(0)

# Γ2
for i in range(10):
    team1, team2, set1, set2 = input('Δώσε ομάδα1(str), ομάδα2(str), σετ1, σετ2 = ')
    if set1 > set2:
        a[team1][0] += 2
        a[team2][0] += 1
    else:
        a[team2][0] += 2
        a[team1][0] += 1
    a[team1][1] += set1
    a[team1][2] += set2
    a[team2][1] += set2
    a[team2][2] += set1

# dummy data comment_out Γ1 Γ2 before execute
names = ['a', 'b', 'c', 'd', 'e']
a = [[12, 10, 15], [10, 15, 15], [7, 6, 11], [19, 15, 0], [10, 8, 7]]

# Γ3
for i in range(4):
    for j in range(4, i, -1):
        if a[j-1][0] < a[j][0]:
            names[j] , names[j-1] = names[j-1], names[j]
            a[j][0], a[j-1][0] = a[j-1][0], a[j][0]
            a[j][1], a[j-1][1] = a[j-1][1], a[j][1]
            a[j][2], a[j-1][2] = a[j-1][2], a[j][2]
        elif a[j-1][0] == a[j][0]:
            if a[j-1][1] < a[j-1][1]:
                names[j] , names[j-1] = names[j-1], names[j]
                a[j][0], a[j-1][0] = a[j-1][0], a[j][0]
                a[j][1], a[j-1][1] = a[j-1][1], a[j][1]
                a[j][2], a[j-1][2] = a[j-1][2], a[j][2]
# Γ4
for i in range(5):
    print names[i], a[i][0], a[i][1], a[i][2]
```

ΘΕΜΑ Δ_0_2017 (Απουσίες επιμορφούμενων σε σεμινάριο)

Σε ένα σεμινάριο διάρκειας 6 μηνών, τηρούνται απουσίες ανά μήνα για κάθε συμμετέχοντα. Στο σεμινάριο συμμετέχουν 5 επιμορφούμενοι και ο καθένας έχει ένα μοναδικό αλφαριθμητικό κωδικό, που αποθηκεύεται στον πίνακα ΚΩΔ[5]. Οι απουσίες κάθε συμμετέχοντα ανά μήνα σεμιναρίου αποθηκεύονται σε δισδιάστατο πίνακα απουσιών ΑΠ[5,6]. Η γραμματεία τηρεί το σύνολο των απουσιών για τα δύο τρίμηνα του εξαμήνου σε πίνακα ΑΠΤΡ[5,2], όπου η πρώτη στήλη προσδιορίζει το πρώτο τρίμηνο και η δεύτερη το δεύτερο τρίμηνο για κάθε συμμετέχοντα.

Να κατασκευάσετε πρόγραμμα σε ΓΛΩΣΣΑ αποτελούμενο από υποπρογράμματα ως εξής:

Δ1. Διαδικασία ΕΙΣ, που διαβάζει τον κωδικό του κάθε επιμορφούμενου, τις απουσίες του ανά μήνα σεμιναρίου και ενημερώνει τον πίνακα ΚΩΔ και τον πίνακα ΑΠ κατάλληλα (θεωρείστε ότι τα δεδομένα εισάγονται σωστά).

Μονάδες 2

Δ2. Συνάρτηση ΑΝΑΖ, που δέχεται τον κωδικό ενός επιμορφούμενου και τον πίνακα των κωδικών ΚΩΔ και επιστρέφει τον αριθμό της γραμμής που βρίσκεται ο κωδικός που αναζητείται. Αν ο κωδικός δεν βρεθεί, επιστρέφει 0.

Μονάδες 4

Δ3. Συνάρτηση ΣΥΝΑΠ, που υπολογίζει το σύνολο απουσιών για έναν επιμορφούμενο σε ένα τρίμηνο. Η συνάρτηση δέχεται τον αριθμό της γραμμής που προσδιορίζει τον επιμορφούμενο στον πίνακα ΑΠ, τον πίνακα των απουσιών και τον αριθμό του πρώτου μήνα του τριμήνου (για παράδειγμα, 1 για το πρώτο τρίμηνο, 4 για το δεύτερο τρίμηνο) και επιστρέφει το σύνολο των απουσιών του τριμήνου.

Μονάδες 3

Δ4. Κύριο πρόγραμμα το οποίο:

α) περιέχει τμήμα δηλώσεων. (μονάδα 1)

β) καλεί τη διαδικασία ΕΙΣ για είσοδο δεδομένων. (μονάδα 1)

γ) για κάθε επιμορφούμενο υπολογίζει το σύνολο των απουσιών των δύο τριμήνων καλώντας τη συνάρτηση ΣΥΝΑΠ και ενημερώνει τον πίνακα ΑΠΤΡ. (μονάδες 3)

δ) διαβάζει επαναληπτικά έναν κωδικό. Για τον συγκεκριμένο κωδικό καλείται η συνάρτηση ΑΝΑΖ. Αν ο κωδικός αντιστοιχεί σε επιμορφούμενο, να εμφανίζει κατάλληλο μήνυμα δυνατότητας ή μη συμμετοχής του στις εξετάσεις. Στις εξετάσεις δικαιούνται συμμετοχής οι επιμορφούμενοι που έχουν λιγότερες από 5 απουσίες σε καθένα από τα δύο τρίμηνα. Αν ο κωδικός δεν βρεθεί, εμφανίζει μήνυμα «ΔΕΝ ΒΡΕΘΗΚΕ Ο ΚΩΔΙΚΟΣ». Η διαδικασία επαναλαμβάνεται μέχρι να δοθείως κωδικός η λέξη ΤΕΛΟΣ. (μονάδες 6)

Μονάδες 11

```
# ---------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ---------------------------------------------------------------------
# Όνομα:       2017_0_D
# Περιγραφή:   Απουσίες επιμορφούμενων σε σεμινάριο
# Ενδλύση:     TasosChatzipapadopoulos
# Python vs:   2.7.10
# Copyright:   (c) 2018
# Url:         http://users.sch.gr/chatzipap
# ---------------------------------------------------------------------

# Δ2
def find(key, code):
```

```
    # key = κωδικός σπουδαστών
    # code = λίστα κωδικών σπουδαστών
    pos = -1
    i = 0
    while pos == -1 and i <= len(code)-1:
        if key == code[i]:
            pos = i
        else:
            i += 1
    return pos


# Δ3
def sum_ap(pos, ap, semester):
    if semester == 1:
        return ap[pos][0] + ap[pos][1] + ap[pos][2]
    elif semester == 2:
        return ap[pos][3] + ap[pos][4] + ap[pos][5]

# Δ1, Δ4 β
code = []
ap = []
for i in range(5):
    code.append(raw_input('Δώσε κωδικό υποψηφίου = '))
    ap.append([])
    for j in range(6):
        ap[i].append(input('Δώσε απουσίες = '))

# Δ4 γ
ap_tr = []
for i in range(5):
    ap_tr.append([])
    ap_tr[i].append(sum_ap(i, ap, 1))
    ap_tr[i].append(sum_ap(i, ap, 2))

# Δ4 δ
st_code = raw_input('Δώσε κωδικό υποψηφίου(END to finish) = ')
while st_code.upper() != 'END':
    pos = find(st_code, code)
    if pos != -1:
        print 'Υπαρκτός κωδικός'
        ap1 = ap_tr[pos][0]
        ap2 = ap_tr[pos][1]
        print ap1, ap2,
        if ap1 < 10 and ap2 < 10:
            print st_code, 'Μπορεί να συμμετέχει στις εξετάσεις'
        else:
            print st_code, 'ΔΕΝ μπορεί να συμμετέχει στις εξετάσεις'
    else:
        print 'ΜΗ υπαρκτός κωδικός'
    st_code = raw_input('Δώσε κωδικό υποψηφίου(END to finish) = ')
```

ΘΕΜΑ Γ_1_2017 (Είσοδος-Έξοδος σε έκθεση απόδημου ελληνισμού)

Σε μια έκθεση αποδήμου ελληνισμού χρησιμοποιείται αίθουσα χωρητικότητας 1000 ατόμων. Στην αίθουσα εγκαταστάθηκε ηλεκτρονικό σύστημα διαχείρισης εισόδου-εξόδου επισκεπτών, το οποίο λειτουργεί ως εξής:

Κάθε φορά που γίνεται είσοδος επισκεπτών εισάγεται η τιμή 1, ενώ κάθε φορά που γίνεται έξοδος επισκεπτών εισάγεται η τιμή 2. Για τον τερματισμό της λειτουργίας του συστήματος εισάγεται η τιμή 0.

Η είσοδος πραγματοποιείται είτε μεμονωμένα είτε σε ομάδες. Προκειμένου να επιτραπεί η είσοδος, ζητείται ο αριθμός επισκεπτών που θέλουν να εισέλθουν και, εφόσον η ενδεχόμενη είσοδός τους δεν υπερβαίνει το όριο χωρητικότητας της αίθουσας, τότε επιτρέπεται· διαφορετικά, απορρίπτεται με κατάλληλο μήνυμα.

Η έξοδος πραγματοποιείται μεμονωμένα, δηλαδή ένα άτομο κάθε φορά. Ο τερματισμός επιτρέπεται, όταν η αίθουσα είναι άδεια.

Για την υποστήριξη του συστήματος να αναπτύξετε πρόγραμμα το οποίο:

Γ1. Να περιλαμβάνει κατάλληλο τμήμα δηλώσεων.

Μονάδες 2

Γ2. Να διαβάζει τον κωδικό επιθυμητής λειτουργίας (1 για είσοδο, 2 για έξοδο και 0 για τερματισμό), μέχρι τον τερματισμό της λειτουργίας του συστήματος.

Μονάδες 4

Γ3. α. Στην περίπτωση που δοθεί ο κωδικός 1, να διαβάζει τον αριθμό των ατόμων και με τη χρήση της λογικής συνάρτησης ΙΝ να ελέγχει αν επιτρέπεται η είσοδός τους. Αν η είσοδός τους επιτρέπεται, εισέρχονται στην αίθουσα· διαφορετικά, εμφανίζεται το μήνυμα ΔΟΚΙΜΑΣΤΕ ΑΡΓΟΤΕΡΑ. (μονάδες 4)

β. Στην περίπτωση που δοθεί ο κωδικός 2, θεωρείται ότι εξέρχεται ένα άτομο. Η εκτέλεση της συγκεκριμένης λειτουργίας να επιτρέπεται, όταν η αίθουσα δεν είναι κενή· διαφορετικά, να εμφανίζει το μήνυμα ΑΔΥΝΑΤΗ ΛΕΙΤΟΥΡΓΙΑ. (μονάδες 2)

Μονάδες 6

Γ4. Μετά τον τερματισμό να εμφανίζει τον συνολικό αριθμό των επισκεπτών, καθώς και το πλήθος των ατόμων της μεγαλύτερης ομάδας που απορρίφθηκε, ή να εμφανίζει το μήνυμα ΔΕΝ ΑΠΟΡΡΙΦΘΗΚΕ ΚΑΜΙΑ ΟΜΑΔΑ.

Μονάδες 4

Γ5. Να αναπτύξετε τη λογική συνάρτηση ΙΝ.

Μονάδες 4

(Να θεωρήσετε ότι δεν απαιτείται έλεγχος εγκυρότητας για τις τιμές εισόδου και ότι η αίθουσα είναι αρχικά κενή).

```
# ----------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ----------------------------------------------------------------------
# Όνομα:        2017_1_C
# Περιγραφή:    Είσοδος-Έξοδος σε έκθεση απόδημου ελληνισμού
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ----------------------------------------------------------------------

# Γ5
def check_in(num, total):
    if num <= 1000 - total:
        return True
    else:
        return False

# Γ2
fail = 0
total = 0
max_fail = 0
code = input('Δώσε κωδικό (1)Είσοδο, (2)Έξοδο, (0)Τέλος = ')
while code != 0:

    # Γ3 α
    if code == 1:
        num_people = input('Πόσα άτομα θα εισέλθουν = ')
        if check_in(num_people, total):
            total += num_people
            print 'Μπορείτε να εισέλθετε στην αίθουσα'
        else:
            fail += 1
            print 'Δοκιμάστε αργότερα'
            if fail == 1:
                max_fail = num_people
            else:
                if num_people > max_fail:
                    max_fail = num_people
    # Γ3 β
    elif code == 2:
        if total != 0:
            total -= 1
        else:
            print 'Αδύνατη Λειτουργία'

    code = input('Δώσε κωδικό (1)Είσοδο, (2)Έξοδο, (0)Τέλος = ')

# Γ4
print 'Συνολικός Αριθμός Επισκεπτών = ', total
if fail == 0:
    print 'Δεν απορρίφθηκε καμία ομάδα'
else:
    print 'Μεγαλύτερη ομάδα που απορρίφθηκε =', max_fail
```

ΘΕΜΑ Δ_1_2017 (Βαθμολογία Εργασιών σε φεστιβάλ ψηφιακής δημιουργίας)

Στο τελευταίο φεστιβάλ ψηφιακής δημιουργίας συμμετείχαν 10 ομάδες μαθητών.Κάθε ομάδα παρουσίασε μια εργασία. Από κάθε ομάδα ζητήθηκε να βαθμολογήσει όλες τις εργασίες, τόσο τη δική της όσο και των υπολοίπων 9 ομάδων.

Να κατασκευάσετε πρόγραμμα το οποίο:

Δ1. Να περιλαμβάνει κατάλληλο τμήμα δηλώσεων.

Μονάδες 2

Δ2. Να καταχωρίζει:

α. τα ονόματα των ομάδων, σε πίνακα Ο[10].(μονάδες 2)

β. τους ακέραιους βαθμούς, σε πίνακα Β[10,10]. Οι βαθμοί να εισάγονται, για κάθε ομάδα με τη σειρά, από την πρώτη μέχρι τη δέκατη, ως εξής:

– να εισάγεται πρώτα ο βαθμός που έδωσε στη δική της εργασία.

– για καθεμιά από τις υπόλοιπες ομάδες, με τη σειρά, που έχουν καταχωριστεί στον πίνακα Ο, να εμφανίζεται το όνομά της και να εισάγεται ο αντίστοιχος βαθμός. (μονάδες 4)

Μονάδες 6

Δ3. Να εμφανίζει το όνομα της ομάδας που συγκέντρωσε τον μεγαλύτερο μέσο όρο βαθμολογίας. Κατά τον υπολογισμό του μέσου όρου να εξαιρούνται ο μεγαλύτερος και ο μικρότερος βαθμός της.

Μονάδες 5

Δ4. Να εμφανίζει το όνομα της ομάδας η οποία βαθμολόγησε τον εαυτό της πλησιέστερα στον μέσο όρο των βαθμών που έλαβε από τις υπόλοιπες ομάδες.

Μονάδες 7

(Για το ερώτημα Δ3 να θεωρήσετε ότι οι τιμές του μέσου όρου, του μικρότερου και του μεγαλύτερου βαθμού είναι μοναδικές. Για το ερώτημα Δ4 να θεωρήσετε ότι η τιμή του μέσου όρου είναι μοναδική).

```
# ----------------------------------------------------------------------
# !/usr/bin/python
# -*- coding: utf-8 -*-
# ----------------------------------------------------------------------
# Όνομα:        2017_1_D
# Περιγραφή:    Βαθμολογία Εργασιών σε φεστιβάλ ψηφιακής δημιουργίας
# Ενδλύση:      TasosChatzipapadopoulos
# Python vs:    2.7.10
# Copyright:    (c) 2018
# Url:          http://users.sch.gr/chatzipap
# ----------------------------------------------------------------------

# Δ2 α, β
names = []

#  Αρχικοποίηση λίστας με δύο διαστάσεις με list comprehension
grades = [[None for i in range(10)] for j in range(10)]

# Ή

# Αρχικοποίηση λίστας χωρίς list comprehension
grades = []
for i in range(10):
```

```
    grades.append([])
    for j in range(10):
        grades[i].append(None)

# Εισαγωγή δεδομένων
for i in range(10):
    names.append(raw_input('Δώσε το όνομα της ομάδας = '))
    grades[i][i] = input('Δώστε το βαθμό της ομάδας σας = ')
    for j in range(10):
        if j <> i:
            grades[i][j] = input('Δώστε το βαθμό της ομάδας = ')

# Δ3
max_mo = -1
imax_mo = -1
for j in range(10):
    xsum = 0
    xmax = grades[i][1]
    xmin = grades[i][1]
    for i in range(10):
        xsum += grades[i][j]
        if xmax < grades[i][j]:
            xmax = grades[i][j]
        if xmin > grades[i][j]:
            xmin = grades[i][j]
        mo = (xsum - xmax - xmin)/ 8.0
        if mo > max_mo:
            max_mo = mo
            imax_mo = i

print names[imax_mo]

# Δ4
# Εύρεση ΜΟ βαθμολογιών από άλλες ομάδες
average = []
for i in range(10):
    xsum = 0
    for j in range(10):
        if i <>j:
            xsum += grades[i][j]
    average.append(xsum / 9.0)

xmin = abs(grades[0][0] - average[0])
ixmin = 0

# Εύρεση κοντινότερης βαθμολογίας στο ΜΟ
for i in range(1,10):
    if abs(grades[i][i] - average[i]) < xmin:
        xmin = abs(grades[i][i] - average[i])
        ixmin = i

print names[ixmin]
```

Αποποίηση ευθύνης

Παρά το γεγονός ότι έγινε έλεγχος των αλγορίθμων των λύσεων, καμιά εγγύηση εξασφάλισης ως προς την πλήρη ορθότητά τους δεν μπορεί να δοθεί. Οι προτάσεις επίλυσης των προβλημάτων δεν έγιναν με στόχο την αλγοριθμική βελτιστοποίησή τους, είναι υποκειμενικές, μπορεί να περιέχουν λάθη και δεν εκφράζουν το σύνολο της επιστημονικής κοινότητας. Διατηρείται το δικαίωμα το περιεχόμενο να αλλάξει, να διαγραφεί προσωρινά ή οριστικά, εξ ολοκλήρου ή εν μέρει ανά πάσα στιγμή και χωρίς προειδοποίηση. Οι αξιώσεις ευθύνης λόγω κάθε είδους ζημιών άυλων ή υλικών που προκύπτουν από τη χρήση του παρόντος αποκλείονται. Δεν φέρεται καμία ευθύνη για αναδημοσιεύσεις του περιεχομένου σε ιστοσελίδες τρίτων μέσω συνδέσμων ή μεταφορτώσεων ή και διανομής με τη μορφή κάθε είδους έντυπου υλικού. Το περιεχόμενο του παρόντος χρησιμοποιείται αποκλειστικά με δική σας ευθύνη.

ISBN 978-960-93-9926-5

Αναστάσιος Χατζηπαπαδόπουλος