ΘΕΜΑ Α

Α1

α. Σωστό

β. Λάθος

γ. Σωστό

δ. Λάθος

ε. Σωστό

Α2

1 → γ

2 → β

3 → α

4 → ε

5 → στ

ΘΕΜΑ Β

Β1

1. len(A)
2. N
3. A[j-1]
4. False
5. f

Β2

1. ['MANOLIS', 'MARIA', 'NIKOS', 'LAZAROS']
2. ['MANOLIS', 'MARIA', 'NIKOS']
3. ['MANOLIS', 'MARIA', 'ADAM', 'NIKOS']
4. ['MANOLIS', 'ADAM', 'NIKOS']
5. ['MANOLIS', 'ADAM', 'NIKOS', 'ANNA']

Β3

```
for i in range(1, 4):
    for j in range(5, 0, -1):
        print i, j
```

ΘΕΜΑ Γ

```
#ερώτημα Γ3
f = open('results.txt', 'w')
pl = 0
k = 0
# ερώτημα Γ1
name = raw_input('Δώσε όνομα: ')
while name != 'ΤΕΛΟΣ':
    # ερώτημα Γ2
    s = 0
    for i in range(5):
        vathmos = int(input('Δώσε βαθμολογία: '))
```

```
        s = s + vathmos
    mo = float(s) / 5
    print mo
    #ερώτημα Γ3
    if mo > 7:
        f.write(name + '\n')
    else:
        #ερώτημα Γ4
        k = k + 1
    pl = pl + 1
    name = raw_input('Δώσε όνομα: ')
#ερώτημα Γ4
pososto = (float(k)/pl)*100
print pososto
f.close()
```

ΘΕΜΑ Δ

```
#συνάρτηση για το ερώτημα Δ3
def MEGISTOS(SALES, TITLES):
    max_pol = SALES[0]
    max_on = TITLES[0]
    for i in range(1, len(SALES)):
        if SALES[i] > max_pol:
            max_pol = SALES[i]
            max_on = TITLES[i]
    return max_on
```

```
TITLES = []
SALES = []
#ερώτημα Δ1
for i in range(40):
    title = raw_input('Δώσε τίτλο βιβλίου: ')
    TITLES.append(title)

    poliseis = int(raw_input('Δώσε αριθμό πωλήσεων: '))
    while poliseis < 0:
        poliseis = int(raw_input('Δώσε ξανά αριθμό πωλήσεων
γιατί δεν ήταν σωστός: '))
    SALES.append(poliseis)

#ερώτημα Δ2
athr = 0
for i in range(40):
    athr = athr + SALES[i]
mo = float(athr) / 40
print athr
print mo

#ερώτημα Δ3
max_tit = MEGISTOS(SALES, TITLES)
print 'Ο τίτλος του βιβλίου με τις περισσότερες πωλήσεις
είναι: ', max_tit
```

```
#ερώτημα Δ4
N = 40
for i in range(N-1):
    for j in range(N-1, i, -1):
        if TITLES[j] < TITLES[j-1]:
            temp = TITLES[j]
            TITLES[j] = TITLES[j-1]
            TITLES[j-1] = temp

            temp2 = SALES[j]
            SALES[j] = SALES[j-1]
            SALES[j-1] = temp2

for i in range(40):
    print TITLES[i], SALES[i]
```

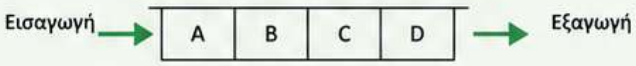
ΘΕΜΑ Α – Οδηγός Απαντήσεων

Α1 – Αλήθεια/Ψευδές προτάσεων

α.	Η λειτουργία της ουράς είναι γνωστή ως FIFO.	ΣΩΣΤΟ
β.	Η έκφραση word[a:b] μας επιστρέφει το τμήμα της συμβολοσειράς ή της λίστας από το στοιχείο word[a] μέχρι και το στοιχείο word[b].	ΛΑΘΟΣ
γ.	Οι τοπικές μεταβλητές ισχύουν για το υποπρόγραμμα στο οποίο δηλώθηκαν.	ΣΩΣΤΟ
δ.	Η ύπαρξη μίας κλάσης σημαίνει και την αυτόματη ύπαρξη ενός αντικειμένου αυτής της κλάσης.	ΛΑΘΟΣ
ε.	Ο τελεστής + όταν εφαρμόζεται σε αντικείμενα τύπου string, έχει ως αποτέλεσμα τη συνένωσή τους σε μία συμβολοσειρά.	ΣΩΣΤΟ

Ουρά (Queue): FIFO = First In First Out

Το πρώτο στοιχείο που εισάγεται είναι και το πρώτο που εξάγεται.



Σωστός κανόνας slicing:

word[a:b] → από τη θέση a μέχρι τη θέση b-1 (το b δεν περιλαμβάνεται).

Παράδειγμα: `word = "Python"`

word[1:4] επιστρέφει `'yth'` (θέσεις 1,2,3) και **ΟΧΙ** μέχρι και το 4.

Γενικά: αρχή (a) περιλαμβάνεται, τέλος (b) δεν περιλαμβάνεται.

Τοπική μεταβλητή (local variable):

Ισχύει μόνο μέσα στο υποπρόγραμμα (συνάρτηση/διαδικασία) στο οποίο δηλώθηκε. Δεν είναι προσβάσιμη εκτός αυτού.

Κλάση (class):

Η κλάση είναι το "σχέδιο" (τύπος). Τα αντικείμενα (instances) δημιουργούνται μόνο όταν το ζητήσουμε.

Παράδειγμα (Python):

```
cless Person:
    pass
p = Person()  # εδώ δημιουργείται αντικείμενο
```

Συνένωση συμβολοσειρών:

Ο τελεστής + ενώνει (concatenete) δύο strings.

Παράδειγμα: `'Καλή' + 'μέρα'` → `'Καλήμέρα'`

Α2 – Αντιστοίχιση εκφράσεων

	ΣΤΗΛΗ Α (Εκφράσεις)	Τι υπολογίζουν
1.	`pow(4,2)`	4 στη δύναμη 2 = **16**
2.	`divmod(2,5)`	Επιστρέφει πηλίκο και υπόλοιπο (2 ÷ 5) = **(0, 2)**
3.	`(5<2) or (4>=3)`	(5<2) → False (4>=3) → True False or True → True
4.	`not(7!=4+5) and (5>3)`	(7!=4+5) → (7!=9) → True not(True) → False (5>3) → True False and True → False
5.	`len('Python')`	Μέτρηση χαρακτήρων της συμβολοσειράς 'Python' = **6**

	ΣΤΗΛΗ Β (Αποτελέσματα)
α.	True
β.	(0,2)
γ.	16
δ.	(2,0)
ε.	False
στ.	6

Αντιστοίχιση

1 → γ (pow(4,2) = 16)

2 → β (divmod(2,5) = (0,2))

3 → α ((5<2) or (4>=3) = True)

4 → ε (not(7!=4+5) and (5>3) = False)

5 → στ (len('Python') = 6)

Υπενθύμιση Χρήσιμων Γνώσεων

- **pow**(x, y): υπολογίζει x^y
- **divmod**(a, b): επιστρέφει (a//b, a%b)
- **or**: αληθής αν τουλάχιστον μία συνθήκη είναι True
- **and**: αληθής μόνο αν όλες οι συνθήκες είναι True
- **not**: αντιστρέφει την τιμή (True ↔ False)
- **len**(s): επιστρέφει το πλήθος χαρακτήρων της συμβολοσειράς s

Σημαντικά για Python

- Η αρίθμηση σε συμβολοσειρές/λίστες ξεκινά από 0.
- Το αποτέλεσμα των λογικών εκφράσεων είναι True ή False.
- Οι σχέσεις επιστρέφουν True (Αληθές) ή False (Ψευδές).

Γρήγορα Παραδείγματα

`pow(4,2)`	→ 16
`divmod(2,5)`	→ (0, 2)
`(5<2) or (4>=3)`	→ False or True → True
`not(7!=4+5) and (5>3)`	→ not(True) and True → False
`len('Python')`	→ 6

Σωστές Απαντήσεις

Α1: α Σ, β Λ, γ Σ, δ Λ, ε Σ

Α2: 1→γ, 2→β, 3→α, 4→ε, 5→στ

ΘΕΜΑ Β – Οδηγός Απαντήσεων

Β1. Τι επιστρέφει κάθε έκφραση;

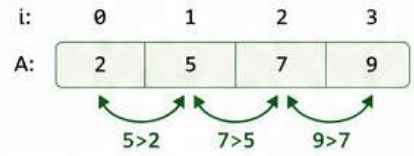
```
def CheckTax(A):
    N = len(A)
    f = True
    for i in range(1, N):
        if A[i] <= A[i-1]:
            f = False
            break
    return f
```

Τι κάνει η συνάρτηση CheckTax(A):

- Ελέγχει αν η λίστα A είναι ταξινομημένη κατά αύξουσα σειρά.
- Ξεκινά με f = True.
- Συγκρίνει κάθε στοιχείο με το προηγούμενό του.
- Αν βρει A[i] <= A[i-1], τότε η λίστα ΔΕΝ είναι σε αύξουσα σειρά θέτει f = False και επιστρέφει False.
- Αν δεν βρει κανένα τέτοιο στοιχείο, επιστρέφει True.

Παράδειγμα λειτουργίας

A = [2, 5, 7, 9]



Όλες οι συγκρίσεις είναι σωστές
άρα f παραμένει True → επιστρέφεται True

Επεξήγηση εκφράσεων

1. len(A) → επιστρέφει το πλήθος των στοιχείων της λίστας A.
2. N → είναι η μεταβλητή που παίρνει την τιμή len(A) (άρα το πλήθος των στοιχείων).
3. A[j-1] → είναι το στοιχείο της λίστας στη θέση j-1.
4. False → λογική τιμή (η συνάρτηση μπορεί να επιστρέψει False).
5. f → η λογική μεταβλητή που δείχνει αν η λίστα είναι ταξινομημένη.

Απαντήσεις Β1

1.	2.	3.	4.	5.
len(A)	N	A[j-1]	False	f

Β2. Αποτέλεσμα εντολών προγράμματος

Λίστες στη Python

- Οι λίστες είναι μεταβλητές που μπορούν να αποθηκεύουν πολλά στοιχεία.
- Τα στοιχεία σε μια λίστα μπαίνουν με append(x) στο τέλος της λίστας.
- Οι θέσεις στη λίστα ξεκινούν από 0.
- insert(θέση, x): εισάγει το x στη θέση και μετακινούνται δεξιά τα επόμενα στοιχεία.
- pop(): αφαιρεί και επιστρέφει το τελευταίο στοιχείο.
- pop(i): αφαιρεί και επιστρέφει το στοιχείο στη θέση i.

Οι εντολές που θα χρησιμοποιηθούν

- append(x) → προσθέτει x στο τέλος
- insert(i, x) → εισάγει x στη θέση i.
- pop() → αφαιρεί και επιστρέφει το τελευταίο στοιχείο.
- pop(i) → αφαιρεί και επιστρέφει το στοιχείο στη θέση i.

Παράδειγμα για insert(2, 'ADAM')

['MANOLIS', 'MARIA', 'NIKOS'] ↓ ['MANOLIS', 'MARIA', 'ADAM', 'NIKOS']

	Εντολή	Αποτέλεσμα που εμφανίζεται
(1)	on = ['MARIA', 'NIKOS', 'LAZAROS'] on = ['MANOLIS'] + on print on	['MANOLIS', 'MARIA', 'NIKOS', 'LAZAROS']
(2)	on.pop() print on	['MANOLIS', 'MARIA', 'NIKOS']
(3)	on.insert(2, 'ADAM') print on	['MANOLIS', 'MARIA', 'ADAM', 'NIKOS']
(4)	on.pop(1) print on	['MANOLIS', 'ADAM', 'NIKOS']
(5)	on = on + ['ANNA'] print on	['MANOLIS', 'ADAM', 'NIKOS', 'ANNA']

Απαντήσεις Β2

1.	2.	3.	4.	5.
['MANOLIS', 'MARIA', 'NIKOS', 'LAZAROS']	['MANOLIS', 'MARIA', 'NIKOS']	['MANOLIS', 'MARIA', 'ADAM', 'NIKOS']	['MANOLIS', 'ADAM', 'NIKOS']	['MANOLIS', 'ADAM', 'NIKOS', 'ANNA']

Β3. Να αντικαταστήσετε τις εντολές while με for έτσι ώστε να εμφανίζεται το ίδιο αποτέλεσμα.

Αρχικός κώδικας (με while)

```
i = 1
while i <= 3:
    j = 5
    while j >= 1:
        print i, j
        j = j - 1
    i = i + 1
```

Ισοδύναμος κώδικας (με for)

```
for i in range(1, 4):
    for j in range(5, 0, -1):
        print i, j
```

Τι πρέπει να γνωρίζετε

- while i <= 3 ↔ for i in range(1, 4) (1, 2, 3)
- while j >= 1 ↔ for j in range(5, 0, -1) (5, 4, 3, 2, 1)

range(start, stop, step)
- start : πρώτη τιμή
- stop : δεν περιλαμβάνεται
- step : βήμα μεταβολής

Παράδειγμα: range(5, 0, -1) → 5, 4, 3, 2, 1

Συχνό λάθος

for j in range(5, 1, -1) παράγει: 5, 4, 3, 2 (λείπει το 1) ✗

Σωστό

for j in range(5, 0, -1) παράγει: 5, 4, 3, 2, 1 ✓

Αποτέλεσμα εκτέλεσης (ίδιο και με τους δύο κώδικες)

1 5 1 4 1 3 1 2 1 1 2 5 2 4 2 3 2 2 2 1 3 5 3 4 3 2 3 1

Τι να θυμάστε

- ✓ Οι λίστες ξεκινούν από θέση 0.
- ✓ append(x) προσθέτει στο τέλος.
- ✓ insert(i, x) εισάγει στη θέση i.
- ✓ pop() αφαιρεί το τελευταίο στοιχείο.
- ✓ pop(i) αφαιρεί το στοιχείο στη θέση i.

Συχνά λάθη

- ✗ Ξεχνάμε ότι η αρίθμηση ξεκινά από 0.
- ✗ [illegible]
- ✗ Νομίζουμε ότι το pop() αφαιρεί από αρχή (ενώ αφαιρεί από το τέλος).

Συνοπτικός Πίνακας Απαντήσεων

- **Β1:** 1 → len(A), 2 → N, 3 → A[j-1], 4 → False, 5 → f
- **Β2:** [illegible] "Απαντήσεις Β2"
- **Β3:** βλέπε μετατροπή while → for

python

ΘΕΜΑ Γ – Οδηγός Απαντήσεων

Οι 5 κριτές βαθμολογούν τους διαγωνιζόμενους με βαθμούς 1 έως 10. Η τελική βαθμολογία κάθε διαγωνιζόμενου προκύπτει από τον μέσο όρο των 5 βαθμών.

ΛΥΣΗ – ΚΩΔΙΚΑΣ PYTHON

```
#ερώτημα Γ3
f = open('results.txt', 'w')
pl = 0
k = 0
# ερώτημα Γ1
name = raw_input('Δώσε ένομα: ')
while name != 'ΤΕΛΟΣ':
   # ερώτημα Γ2
   s = 0
   for i in range(5):
       vathmos = int(input('Δώσε βαθμολογία: '))
       s = s + vathmos
   mo = float(s) / 5
   print mo
   #ερώτημα Γ3
   if mo > 7:
        f.write(name + '\n')
   else:
       #ερώτημα Γ4
        k = k + 1
   pl = pl + 1
    name = raw_input('Δώσε ένομα: ')
#ερώτημα Γ4
pososto = (float(k)/pl)*100
print pososto
f.close()
```

ΤΙ ΚΑΝΕΙ ΤΟ ΠΡΟΓΡΑΜΜΑ

Γ1
- Διαβάζει το όνομα κάθε διαγωνιζόμενου.
- Επαναλαμβάνει τη διαδικασία όσο το όνομα ΔΕΝ είναι 'ΤΕΛΟΣ'.

Γ2
- Αρχικοποιεί το άθροισμα βαθμών `s = 0`.
- Διαβάζει 5 βαθμολογίες (από τους 5 κριτές) και τις προσθέτει στο `s`.
- Υπολογίζει τον μέσο όρο: `mo = s / 5`.
- Εμφανίζει τον μέσο όρο.

Γ3
Αν ο μέσος όρος `mo > 7`:
γράφει το όνομα του διαγωνιζόμενου στο αρχείο 'results.txt' (ένας/μία ανά γραμμή).

Γ4
Αν ο μέσος όρος `mo ≤ 7`:
αυξάνει τον μετρητή `k` (αριθμός διαγωνιζόμενων με βαθμολογία ≤ 7).

Σε κάθε επανάληψη αυξάνει τον μετρητή `pl` (συνολικός αριθμός διαγωνιζόμενων).

Μετά το τέλος (όταν δοθεί 'ΤΕΛΟΣ'):
υπολογίζει και εμφανίζει το ποσοστό (%) των διαγωνιζόμενων με βαθμολογία ≤ 7, δηλαδή `(k / pl) * 100`.

ΑΝΑΛΥΣΗ ΜΕΤΑΒΛΗΤΩΝ

Μεταβλητή	Τύπος	Περιγραφή
f	file	Το αρχείο 'results.txt' (για εγγραφή)
pl	int	Πλήθος όλων των διαγωνιζόμενων
k	int	Πλήθος διαγωνιζόμενων με mo ≤ 7
name	str	Το όνομα του διαγωνιζόμενου
s	int	Άθροισμα βαθμών των 5 κριτών
vathmos	int	Κάθε βαθμολογία κριτή (1–10)
mo	float	Μέσος όρος (τελική βαθμολογία)
pososto	float	Ποσοστό (%) διαγωνιζόμενων με mo ≤ 7

ΠΑΡΑΔΕΙΓΜΑ ΕΚΤΕΛΕΣΗΣ

Όνομα διαγωνιζόμενου	Βαθμολογίες κριτών	Μέσος όρος (mo)	mo > 7;	Ενέργεια
ΜΑΡΙΑ	8 7 9 6 8	7.6	ΝΑΙ	γράφεται στο αρχείο
ΝΙΚΟΣ	6 5 6 5 6	5.6	ΟΧΙ	k = 1
ΑΝΝΑ	9 9 8 9 10	9.0	ΝΑΙ	γράφεται στο αρχείο
ΠΑΝΟΣ	7 7 7 6 6	6.6	ΟΧΙ	k = 2
ΤΕΛΟΣ	–	–	–	τέλος εισαγωγής

ΑΡΧΕΙΟ results.txt (μετά το τέλος)

ΜΑΡΙΑ
ΑΝΝΑ

ΥΠΟΛΟΓΙΣΜΟΣ ΠΟΣΟΣΤΟΥ

`k = 2` (διαγωνιζόμενοι με mo ≤ 7)
`pl = 4` (σύνολο διαγωνιζόμενων)
`pososto = (2 / 4) * 100 = 50.0`
Εμφανίζεται: **50.0**

ΤΙ ΝΑ ΘΥΜΑΣΤΕ

- raw_input(): διαβάζει αλφαριθμητική τιμή (string).
- input() → int(input(...)): διαβάζουμε αριθμό.
- Άθροισμα 5 βαθμών και μέσος όρος: mo = s / 5.
- Αν mo > 7 → γράφουμε το όνομα στο αρχείο.
- Αν mo ≤ 7 → αυξάνουμε τον μετρητή k.
- Υπολογίζουμε το ποσοστό στο τέλος: (k / pl) * 100.

ΣΥΧΝΑ ΛΑΘΗ

- ✗ Ξεχνάμε να μετατρέψουμε την είσοδο βαθμολογίας σε int.
- ✗ Ξεχνάμε να αρχικοποιήσουμε `s = 0` σε κάθε διαγωνιζόμενο.
- ✗ Ξεχνάμε να αυξήσουμε τον μετρητή pl.
- ✗ Υπολογίζουμε το ποσοστό πριν δοθεί 'ΤΕΛΟΣ'.

ΣΥΝΟΠΤΙΚΕΣ ΑΠΑΝΤΗΣΕΙΣ ΣΤΑ ΕΡΩΤΗΜΑΤΑ

Γ1 α) `name = raw_input('Δώσε όνομα: ')` `while name != 'ΤΕΛΟΣ':`
Γ2 `for i in range(5):` `vathmos = int(input('Δώσε βαθμολογία: '))` `s = s + vathmos` `mo = float(s) / 5` `print mo`
Γ3 `if mo > 7:` `f.write(name + '\n')`
Γ4 `pososto = (float(k)/pl)*100` `print pososto`

ΘΕΜΑ Δ – Οδηγός Απαντήσεων

ΛΥΣΗ – ΚΩΔΙΚΑΣ PYTHON

```
# συνάρτηση για το ερώτημα Δ3
def MEGISTOS(SALES, TITLES):
    max_pol = SALES[0]
    max_on = TITLES[0]
    for i in range(1, len(SALES)):
        if SALES[i] > max_pol:
            max_pol = SALES[i]
            max_on = TITLES[i]
    return max_on

TITLES = []
SALES = []
# ερώτημα Δ1
for i in range(40):
    title = raw_input('Δώσε τίτλο βιβλίου: ')
    TITLES.append(title)

    poliseis = int(raw_input('Δώσε αριθμό πωλήσεων: '))
    while poliseis < 0:
        poliseis = int(raw_input('Δώσε ξανά αριθμό
                  πωλήσεων γιατί δεν ήταν σωστός: '))
    SALES.append(poliseis)

# ερώτημα Δ2
athr = 0
for i in range(40):
    athr = athr + SALES[i]
mo = float(athr) / 40
print athr
print mo

# ερώτημα Δ3
max_tit = MEGISTOS(SALES, TITLES)
print 'Ο τίτλος του βιβλίου με τις περισσότερες
      πωλήσεις είναι: ', max_tit

# ερώτημα Δ4
N = 40
for i in range(N-1):
    for j in range(N-1, i, -1):
        if TITLES[j] < TITLES[j-1]:
            temp = TITLES[j]
            TITLES[j] = TITLES[j-1]
            TITLES[j-1] = temp

            temp2 = SALES[j]
            SALES[j] = SALES[j-1]
            SALES[j-1] = temp2

for i in range(40):
    print TITLES[i], SALES[i]
```

ΤΙ ΖΗΤΟΥΝ ΤΑ ΕΡΩΤΗΜΑΤΑ

Δ1 Να διαβάσει τον τίτλο κάθε βιβλίου και να τον καταχωρίζει σε μία λίστα με όνομα TITLES και τον αριθμό πωλήσεων του κάθε βιβλίου (έλεγχος εγκυρότητας: δεν επιτρέπονται αρνητικές τιμές) και να τον καταχωρίζει σε μία λίστα με όνομα SALES.

Δ2 Να υπολογίσει και να εμφανίσει:
- τις συνολικές πωλήσεις όλων των βιβλίων
- τον μέσο όρο των πωλήσεων όλων των βιβλίων

Δ3 Να καλέσει συνάρτηση MEGISTOS(), η οποία θα δέχεται τις λίστες SALES και TITLES, θα υπολογίζει και θα επιστρέφει τον τίτλο του βιβλίου με τις περισσότερες πωλήσεις και να τον εμφανίζει με το μήνυμα:
"Ο τίτλος του βιβλίου με τις περισσότερες πωλήσεις είναι: ".

Σημείωση:
Θεωρήστε ότι υπάρχει ένας μόνο τίτλος βιβλίου με μέγιστο αριθμό πωλήσεων.

Δ4 Να ταξινομεί τη λίστα TITLES σε αλφαβητική σειρά με χρήση του αλγόριθμου ταξινόμησης ευθείας ανταλλαγής (φυσαλίδα – bubble sort), αναδιατάσσοντας συγχρόνως τη λίστα SALES. Στο τέλος, να εμφανίζει τις λίστες TITLES και SALES.

ΤΙ ΚΑΝΕΙ ΚΑΘΕ ΤΜΗΜΑ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ

Συνάρτηση MEGISTOS(SALES, TITLES)	Επιστρέφει τον τίτλο του βιβλίου με τις περισσότερες πωλήσεις (μέγιστη τιμή στη λίστα SALES).
Ερώτημα Δ1	Διαβάζει 40 τίτλους και τους αποθηκεύει στη λίστα TITLES. Διαβάζει 40 αριθμούς πωλήσεων (έλεγχος: όχι αρνητικοί) και τους αποθηκεύει στη λίστα SALES.
Ερώτημα Δ2	Υπολογίζει το άθροισμα των πωλήσεων (athr) και τον μέσο όρο (mo = athr / 40) και τους εμφανίζει.
Ερώτημα Δ3	Καλεί τη συνάρτηση MEGISTOS και εμφανίζει το όνομα του βιβλίου με τις περισσότερες πωλήσεις.
Ερώτημα Δ4	Ταξινομεί αλφαβητικά τη λίστα TITLES με bubble sort και σε κάθε ανταλλαγή τίτλων κάνει και την ίδια ανταλλαγή στους αντίστοιχους αριθμούς της λίστας SALES. Τέλος, εμφανίζει τις τελικές λίστες TITLES και SALES.

BUBBLE SORT (ΕΥΘΕΙΑ ΑΝΤΑΛΛΑΓΗ)

- Συγκρίνουμε γειτονικά στοιχεία και τα ανταλλάσσουμε αν είναι σε λάθος σειρά.
- Επαναλαμβάνουμε μέχρι να ταξινομηθεί όλη η λίστα.
- Κάθε πέρασμα "φουσκώνει" το μεγαλύτερο στοιχείο στο τέλος.

Στο πρόγραμμα:
- Εξωτερικό for: i από 0 μέχρι N-2 (N = 40)
- Εσωτερικό for: j από N-1 μέχρι i+1 με βήμα -1
- Αν TITLES[j] < TITLES[j-1], τότε ανταλλάσσουμε TITLES[j] με TITLES[j-1] και ταυτόχρονα SALES[j] με SALES[j-1]

ΠΑΡΑΔΕΙΓΜΑ ΕΚΤΕΛΕΣΗΣ (μικρό παράδειγμα 5 βιβλίων)

Αρχικά δεδομένα

Δείκτης	0	1	2	3	4
TITLES	Python	Δίκτυα	C++	Αλγόριθμοι	Βάσεις
SALES	15	8	20	12	5

Μετά την ταξινόμηση (αλφαβητικά ως προς TITLES)

Δείκτης	0	1	2	3	4
TITLES	Αλγόριθμοι	Βάσεις	C++	Python	Δίκτυα
SALES	12	5	20	15	8

Αποτέλεσμα ερωτήματος Δ3 (στο παράδειγμα)
Ο τίτλος του βιβλίου με τις περισσότερες πωλήσεις είναι: C++

ΑΝΑΛΥΣΗ ΜΕΤΑΒΛΗΤΩΝ

Μεταβλητή	Τύπος	Περιγραφή
TITLES	list	Λίστα με τους τίτλους των 40 βιβλίων
SALES	list	Λίστα με τους αριθμούς πωλήσεων των 40 βιβλίων
i	int	Μετρητής για τα for
j	int	Μετρητής εσωτερικού for (bubble sort)
title	str	Τίτλος βιβλίου που διαβάζεται
poliseis	int	Αριθμός πωλήσεων που διαβάζεται
athr	int	Σύνολο πωλήσεων όλων των βιβλίων
mo	float	Μέσος όρος πωλήσεων
max_pol	int	Μέγιστος αριθμός πωλήσεων (στη συνάρτηση)
max_on	str	Τίτλος βιβλίου με μέγιστες πωλήσεις

ΕΠΕΞΗΓΗΣΗ ΣΥΝΑΡΤΗΣΗΣ MEGISTOS(SALES, TITLES)

1. Θέτουμε αρχικά ως μέγιστες τιμές το πρώτο στοιχείο των λιστών: max_pol και max_on.
2. Διατρέχουμε τη λίστα SALES από τη θέση 1 μέχρι το τέλος.
3. Αν βρούμε μεγαλύτερη τιμή, ενημερώνουμε το max_pol και αποθηκεύουμε τον αντίστοιχο τίτλο στο max_on.
4. Επιστρέφουμε τον τίτλο (max_on).

ΣΗΜΑΝΤΙΚΕΣ ΠΑΡΑΤΗΡΗΣΕΙΣ

- Διαβάζουμε ακριβώς 40 βιβλία.
- Οι πωλήσεις πρέπει να είναι >= 0.
- Οι δύο λίστες TITLES και SALES έχουν πάντα το ίδιο μήκος και οι αντίστοιχες θέσεις τους συνδέονται μεταξύ τους.
- Το bubble sort διατηρεί τη συσχέτιση τίτλου – πωλήσεων με τις ανταλλαγές.

ΣΥΝΟΠΤΙΚΕΣ ΑΠΑΝΤΗΣΕΙΣ ΣΤΑ ΕΡΩΤΗΜΑΤΑ

Δ1 Διαβάζουμε 40 τίτλους (TITLES) και 40 αριθμούς πωλήσεων (SALES) με έλεγχο εγκυρότητας (όχι αρνητικοί).

Δ2 Υπολογίζουμε το άθροισμα (athr) και τον μέσο όρο (mo = athr / 40) και τα εμφανίζουμε.

Δ3 Καλούμε MEGISTOS(SALES, TITLES) και εμφανίζουμε το όνομα του βιβλίου με τις περισσότερες πωλήσεις.

Δ4 Ταξινομούμε αλφαβητικά τη λίστα TITLES με bubble sort και αναδιατάσσουμε αντίστοιχα τη λίστα SALES.
Στο τέλος εμφανίζουμε τις δύο λίστες.