Σχολή
Ηλεκτρολόγων
Μηχανικών &
Μηχανικών
Υπολογιστών

Εργαλεία Ανάπτυξης Λογισμικού και Προγραμματισμός Συστημάτων

Τίτλος Εργασίας:Εργασία σε Python
Ομάδα Εργασίας :LAB21142503
Ονοματεπώνυμο :Γρηγοριάδης Ιωάννης 2014030007
Ονοματεπώνυμο :Μάνεσης Αθανάσιος 2014030061

Πίνακας Περιεχομένων:

1) Περιγραφή Άσκησης:

Η δεύτερη εργαστηριακή άσκηση για το μάθημα Εργαλεία Ανάπτυξης Λογισμικού και Προγραμματισμός Συστημάτων αφορά την υλοποίηση προγράμματος το οποίο διαβάζει αποδείξεις σε μορφή αρχείων και κάνει υπολογισμούς επί των αποδείξεων. Συγκεκριμένα το πρόγραμμα έχει μενού τεσσάρων επιλογών. Η πρώτη διαβάζει αρχείο με αποδείξεις, η δεύτερη τυπώνει το συνολικό ποσό πωλήσεων του προϊόντος ανά ΑΦΜ, η τρίτη τυπώνει το άθροισμα πωλήσεων ανά προϊόν για συγκεκριμένο ΑΦΜ και η τέταρτη τερματίζει το πρόγραμμα.
Τα αρχεία εισόδου έχουν συγκεκριμένη δομή και το πρόγραμμα πρέπει να κάνει έλεγχο ορθότητας επί αυτών.

2) Διαδικασία επίλυσης:

Το πρόγραμμα υλοποιήθηκε εξολοκλήρου με χρήση ρουτινών (Functions) γι’ αυτό το λόγο κρίνεται σκόπιμο να εξηγηθούν τα Functions υπό μορφή bullet points. Να σημειωθεί ότι σε κάποιες συναρτήσεις εμφανίζεται σαν όρισμα η σταθερά verbose. Αυτή η σταθερά όταν είναι True εμφανίζονται στο πρόγραμμα μηνύματα σφαλμάτων καθώς και ο χρόνος εκτέλεσης των ερωτημάτων της άσκησης. Η σταθερά χρησιμοποιήθηκε για σκοπούς αποσφαλμάτωσης και για χρονικούς έλεγχους του προγράμματος. Καθώς η άσκηση διασαφηνίζει ότι δεν πρέπει να εμφανίζονται καθόλου μηνύματα η σταθερά είναι αρχικοποιημένη στο False και δεν θα γίνει καμία μελλοντική αναφορά σε αυτήν.

- **round_to_float(value):** Παίρνει έναν δεκαδικό αριθμό και τον επιστρέφει στρογγυλεμένο στα δύο ψηφία. Ο κύριος λόγος που χρησιμοποιείται είναι για να διορθώσει προβλήματα που προκύπτουν από πολλαπλασιασμούς δεκαδικών αριθμών. (π.χ. 13.379999999999999 != 13.38). Επίσης χρησιμοποιείται και κατά την εκτύπωση.

- **is_float(value):** Παίρνει μία μεταβλητή αγνώστου τύπου και επιστρέφει True εάν η μεταβλητή ήταν δεκαδικός αριθμός (float). Αρχικά γίνεται προσπάθεια να ανατεθεί η άγνωστη μεταβλητή σε ένα float αρχικοποιημένο στο 0. Εάν αυτό αποτύχει τότε η μεταβλητή ήταν αλφαριθμητικό και επιστρέφει False. Χρησιμοποιείται κατά τον έλεγχο ορθότητας των αποδείξεων

- **is_int(value):** Είναι η αντίστοιχη συνάρτηση για τους ακεραίους αριθμούς. (βλ. is_float(value))

- **check_int_range(low, high, num):** Παίρνει έναν ακέραιο αριθμό καθώς το άνω και κάτω όριο. Η συνάρτηση επιστρέφει 0 εάν ο ακαριαίος αριθμός είναι εντός των δύο ορίων και 1 εάν είναι εκτός. Χρησιμοποιείται για να ελέγξει την είσοδο του χρήστη.

- **is_exist(name, verbose)**: Παίρνει ένα αλφαριθμητικό το οποίο μπορεί να είναι το όνομα ενός αρχείου ή το μονοπάτι (Full path) προς το αρχείο. Κάνει έλεγχο εάν το αρχείο υπάρχει και εφόσον υπάρχει επιστέφει τον δείκτη στο αρχείο με δικαιώματα διαβάσματος.

- **match_in_line(match_of,line)**: Οι αποδείξεις στο αρχείο χωρίζονται με γραμμές που περιέχουν μόνο το χαρακτήρα “-”. Καθώς το αρχείο διαβάζεται, η κάθε γραμμή περνά από αυτή την συνάρτηση με match_of = “-” και ελέγχεται εάν ο κάθε χαρακτήρας της γραμμής είναι ίσος με “-”. Εάν ισχύει τότε επιστρέφει True. Κάθε γραμμή στο τέλος της έχει τον χαρακτήρα διαφυγής “\n” οπότε ο τελευταίος χαρακτήρας εξαιρείται από τον έλεγχο.

- **validate_receipt(receipt, line_counter, file_name, verbose)**: (Τα ορίσματα line_counter και file_name χρησιμοποιηθήκαν για σκοπούς αποσφαλμάτωσης)
  Παίρνει μία απόδειξη (receipt = []) η οποία είναι μία λίστα στην οποία το πρώτο και τελευταίο στοιχείο της είναι ένα αλφαριθμητικό αποτελούμενο από “-”. Στην συνέχεια κάνει ελέγχους ορθότητας για τα επιμέρους στοιχεία της απόδειξης και εφόσον δεν υπάρχουν σφάλματα επιστρέφει 0. Η συνάρτηση κάνει τους εξής ελέγχους ορθότητας στους οποίους εάν κάποια απόδειξη αποτύχει έστω και σε έναν τότε επιστέφει 1:
  - Ελέγχει εάν το μέγεθος της απόδειξης είναι μικρότερο από τρία καθώς κάθε απόδειξη πρέπει να περιέχει τουλάχιστον ένα προϊόν και πρέπει να υπάρχουν το ΑΦΜ και το ΣΥΝΟΛΟ στην πρώτη και τελευταία γραμμή.
  - Ελέγχει εάν η πρώτη και τελευταία γραμμή έχει περισσότερα από δύο στοιχεία καθώς στην πρώτη γραμμή πρέπει να αναγράφεται μόνο το αλφαριθμητικό ΑΦΜ: ακολουθούμενο από ένα νούμερο και στην τελευταία το αλφαριθμητικό ΣΥΝΟΛΟ: ακολουθούμενο από έναν δεκαδικό αριθμό.
  - Ελέγχει εάν το πρώτο στοιχείο της πρώτης γραμμή και το πρώτο στοιχείο της τελευταίας είναι τα αλφαριθμητικά ΑΦΜ: και ΣΥΝΟΛΟ: αντίστοιχα.
  - Ελέγχει εάν το δεύτερο στοιχείο της πρώτης γραμμής είναι ακαριαίος αριθμός καθώς αναπαριστά τον αριθμό ΑΦΜ.
  - Ελέγχει εάν το νούμερο του ΑΦΜ αποτελείται από δέκα ψηφία.
  - Ελέγχει εάν το δεύτερο στοιχείο της τελευταίας γραμμής είναι δεκαδικός αριθμός καθώς αναπαριστά τον ΣΥΝΟΛΟ.
  - Ελέγχει εάν οι γραμμές των προϊόντων αποτελούνται από τρία στοιχεία μετά το ΟΝΟΜΑ_ΠΡΟΪΟΝΤΟΣ καθώς πρέπει να είναι της μορφής ΠΟΣΟΤΗΤΑ, ΤΙΜΗ, ΣΥΝΟΛΙΚΗ_ΤΙΜΗ. Ο λόγος που δεν γίνεται ο έλεγχος για τέσσερα στοιχεία είναι γιατί το όνομα του προϊόντος μπορεί να αποτελείται από περισσότερες από μία λέξεις. (Επίσης δεν γίνεται κανένας έλεγχος για το όνομα του προϊόντος γιατί μπορεί να είναι οτιδήποτε

συμπεριλαμβανομένων αριθμών π.χ 7up, tic-tac, coca cola κ.τ.λ.)
    - Ελέγχει εάν η ποσότητα του προϊόντος είναι ακέραιος αριθμός καθώς η τιμή και το σύνολο είναι δεκαδικός.
    - Ελέγχει εάν ο πολλαπλασιασμός της ποσότητας με την τιμή του προϊόντος ισούται με το σύνολο που αναγράφεται.
    - Ελέγχει εάν τα μερικά σύνολα από όλα τα προϊόντα της απόδειξης ισούνται με το τελικό σύνολο της απόδειξης.

- **fill_the_structure(receipt)**: Παίρνει μία απόδειξη στην οποία η ορθότητα έχει ελεγχθεί και την καταχωρεί σε μία ιδικά διαμορφωμένη δομή δεδομένων η οποία σκοπό έχει να επισπευσθούν οι ευρέσεις για μεγάλο όγκο δεδομένων. Χρησιμοποιώντας string manipulation επί των γραμμών της απόδειξης καταλήγει στην εξής δομή όπου είναι ένα λεξικό αποτελούμενο από τα πεδία ΑΦΜ, ΠΡΟΪΟΝ και ΣΥΝΟΛΟ. Το πεδίο ΠΡΟΪΟΝ είναι ένα λεξικό αποτελούμενο από τα πεδία ΟΝΟΜΑ_ΠΡΟΪΟΝΤΟΣ, ΠΟΣΟΤΗΤΑ, ΤΙΜΗ, ΣΥΝΟΛΙΚΗ_ΤΙΜΗ. Τέλος επειδή κάθε απόδειξη μπορεί να περιέχει περισσότερα από ένα προϊόντα το πεδίο - λεξικό {ΠΡΟΪΟΝ} είναι μέρος μιας λίστας.

  Προϊόν: { ΟΝΟΜΑ_ΠΡΟΪΟΝΤΟΣ, ΠΟΣΟΤΗΤΑ, ΤΙΜΗ, ΣΥΝΟΛΙΚΗ_ΤΙΜΗ }
  Λίστα: [{ Προϊόν1 }, { Προϊόν2 }, ..., { ΠροϊόνΝ }]

  Δομή: { “ΑΦΜ”, [Λίστα], “ΣΥΝΟΛΟ” }

- **create_option_two_list(book, product_name)**: Παίρνει το book[] το οποίο είναι μία λίστα από αποδείξεις της μορφής που δείχνει η συνάρτηση fill_the_structure(receipt) καθώς και το όνομα του προϊόντος στο οποίο θα γίνει η έρευνα. Γίνεται έλεγχος για το αν το βιβλίο έχει τουλάχιστον μία απόδειξη αν όχι τότε επιστρέφει 1. Με χρήση δύο for-loop ελέγχουμε κάθε προϊόν για κάθε ΑΦΜ που υπάρχει στο βιβλίο και αν το όνομα του προϊόντος που έδωσε ο χρήστης ταιριάζει σε κάποιο ΑΦΜ τότε βάζουμε ένα string της μορφής “ΑΦΜ ΣΥΝΟΛΙΚΟ_ΠΟΣΟ” σε μία λίστα. Η συνάρτηση δεν κάνει έλεγχο αν κάποιο ΑΦΜ υπάρχει ήδη στη λίστα οπότε μια πιθανή μορφή της λίστας μπορεί να είναι [“ΑΦΜ1 ΣΥΝΟΛΙΚΟ_ΠΟΣΟ1”, “ΑΦΜ1 ΣΥΝΟΛΙΚΟ_ΠΟΣΟ2”, “ΑΦΜ2 ΣΥΝΟΛΙΚΟ_ΠΟΣΟ1” ...]. Έλεγχος για διπλά ΑΦΜ και ταξινόμηση γίνεται στη συνάρτηση print_option_two_list(list).

- **print_option_two_list(list)**: Παίρνει την λίστα η οποία δημιουργήθηκε στην συνάρτηση create_option_two_list(book, product_name) και την ελέγχει. Εάν είναι κενή τότε επιστρέφει 1. Στη συνέχεια ελέγχει την λίστα για τυχών πολλαπλά ΑΦΜ με τον ίδιο αριθμό και τα αφαιρεί ενημερώνοντας το συνολικό πόσο τους. Αυτό επιτυγχάνεται με τη βοήθεια δύο παραλλήλων λιστών afm[] και total_sum[] όπου τα μοναδικά ΑΦΜ μπαίνουν στην afm[] και το ποσό μπαίνει στην παράλληλη θέση στην λίστα total_sum[]. Όταν βρεθεί ΑΦΜ το οποίο υπάρχει ήδη στην afm[] τότε ενημερώνεται το total_sum[] στην αντίστοιχη

θέση. Τέλος στρογγυλοποιούνται τα ποσά στα δύο ψηφία, ταξινομούνται με βάση το ΑΦΜ και γίνεται εκτύπωση.

- **create_option_three_list(book, afm)**: Παίρνει το book[] το οποίο είναι μία λίστα από αποδείξεις της μορφής που δείχνει η συνάρτηση fill_the_structure(receipt) καθώς και το ΑΦΜ στο οποίο θα γίνει η έρευνα. Γίνεται έλεγχος για το αν το βιβλίο έχει τουλάχιστον μία απόδειξη αν όχι τότε επιστρέφει 1. Με χρήση δύο for-loop ελέγχουμε κάθε ΑΦΜ που υπάρχει στο βιβλίο και αν το ΑΦΜ που έδωσε ο χρήστης ταιριάζει σε κάποιο ΑΦΜ τότε βάζουμε ένα string της μορφής "ΠΡΟΪΟΝ: ΣΥΝΟΛΙΚΟ_ΠΟΣΟ" σε μία λίστα για κάθε προϊόν που έχει το συγκεκριμένο ΑΦΜ. Η συνάρτηση δεν κάνει έλεγχο αν κάποιο προϊόν υπάρχει ήδη στη λίστα οπότε μια πιθανή μορφή της λίστας μπορεί να είναι ["ΠΡΟΪΟΝ1 ΣΥΝΟΛΙΚΟ_ΠΟΣΟ1", " ΠΡΟΪΟΝ1 ΣΥΝΟΛΙΚΟ_ΠΟΣΟ2", " ΠΡΟΪΟΝ2 ΣΥΝΟΛΙΚΟ_ΠΟΣΟ3" ...]. Έλεγχος για διπλά προϊόντα και ταξινόμηση γίνεται στη συνάρτηση print_option_three_list(list).

- **print_option_three_list(list)**: Παίρνει την λίστα η οποία δημιουργήθηκε στην συνάρτηση create_option_three_list(book, afm) και την ελέγχει. Εάν είναι κενή τότε επιστρέφει 1. Στη συνέχεια ελέγχει την λίστα για τυχών πολλαπλά προϊόντα με τον ίδιο όνομα και τα αφαιρεί ενημερώνοντας το συνολικό πόσο τους. Αυτό επιτυγχάνεται με τη βοήθεια δύο παραλλήλων λιστών product[] και total_sum[] όπου τα μοναδικά προϊόντα μπαίνουν στην product[] και το ποσό μπαίνει στην παράλληλη θέση στην λίστα total_sum[]. Όταν βρεθεί προϊόν το οποίο υπάρχει ήδη στην product[] τότε ενημερώνεται το total_sum[] στην αντίστοιχη θέση. Τέλος στρογγυλοποιούνται τα ποσά στα δύο ψηφία, ταξινομούνται με βάση το προϊόν και γίνεται εκτύπωση.

- **option_one(book, verbose)**: Παίρνει το book[] το οποίο είναι μία λίστα από αποδείξεις της μορφής που δείχνει η συνάρτηση fill_the_structure(receipt). Ζητείται από τον χρήστη το όνομα του αρχείου και με την βοήθεια της συνάρτησης is_exist(file_name, verbose) καταχωρεί στην μεταβλητή fp τον δείκτη στο αρχείο εάν αυτό υπάρχει. Εάν δεν υπάρχει τότε επιστρέφει το βιβλίο ως έχει. Στη συνέχεια με τη βοήθεια της συνάρτησης match_in_line('-',line) βρίσκει το σημείο στο οποίο αρχίζουν οι κάθε αποδείξεις και αφού τις ελέγξει για τυχών λάθη κάνοντας χρήση της validate_receipt(to_be_checked_receipt, counter, file_name, verbose) τις αποθηκεύει στο βιβλίο με την κατάλληλη δομή (βλ. fill_the_structure(to_be_checked_receipt)). Η συνάρτηση παίρνει και ανανεώνει το βιβλίο με αυτό το τρόπο βιβλίο δεν επαναγράφεται και μπορούν να μπουν πολλαπλά αρχεία.

- **option_two(book, verbose)**: Αφού πάρει το όνομα του προϊόντος από τον χρήστη καλεί την συνάρτηση print_option_two_list() με όρισμα τη συνάρτηση create_option_two_list().

- **option_three(book, verbose)**: Αφού πάρει το ΑΦΜ από τον χρήστη καλεί την συνάρτηση print_option_three_list() με όρισμα τη συνάρτηση create_option_three_list().

- **option_four()**: Τερματίζει το πρόγραμμα με την χρήση της εντολής exit().

- **main_menu(verbose)**: Τυπώνει στην κονσόλα το σχετικό μήνυμα και ζητά από τον χρήστη την επιλογή του. Η συνάρτηση κάνει έλεγχο για το αν η επιλογή είναι αλφαριθμητικό και εάν είναι ξανατυπώνει το μήνυμα. Επίσης με τη βοήθεια της check_int_range(1,4,option) ελέγχει αν ο αριθμός πλέον είναι μεταξύ των ορίων. Η εκτύπωση του μηνύματος στην οθόνη δεν σταματά εκτός εάν ο χρήστης επιλέξει κάτι εντός των ορίων. Έχουν παρθεί οι κατάλληλες προφυλάξεις για την περίπτωση που ο χρήστης επιλέξει την επιλογή δύο ή τρία πριν εισαγάγει αρχείο στο πρόγραμμα.

- **Main()**: Καλεί την main_menu() και αποθηκεύει την επιλογή του χρήστη στην μεταβλητή option. Έπειτα με τη χρήση if καθοδηγεί το πρόγραμμα στις κατάλληλες συναρτήσεις βάση της επιλογής του χρήστη.

3) Ερωτήματα:

Α) Πώς διαβάζετε κάθε απόδειξη;

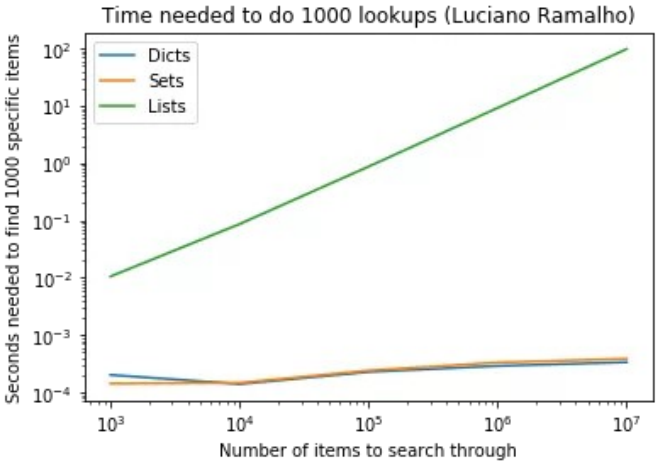
Το διάβασμα κάθε απόδειξης γίνεται στην συνάρτηση option_one(book,verbose). Η συνάρτηση αυτή παίρνει σαν όρισμα και επιστρέφει το βιβλίο με αυτόν το τρόπο το πρόγραμμα μπορεί να δέχεται πολλαπλά αρχεία με αποδείξεις επειδή το επόμενο αρχείο γράφεται στο τέλος του βιβλίου χωρίς αυτό να ξανά αρχικοποιείται. Αφού βεβαιωθούμε ότι το αρχείο που έδωσε ο χρήστης υπάρχει με τη χρήση της συνάρτησης is_exist(file_name, verbose) αρχίζουμε το διάβασμα του αρχείου γραμμή προς γραμμή έως ότου να βρούμε την πρώτη γραμμή στην οποία υπάρχει μόνο ο χαρακτήρας “-”. Αυτό το επιτυγχάνουμε με την μεταβλητή mode η οποία είναι αρχικοποιημένη στο off και όταν το πρόγραμμά βρει τη πρώτη γραμμή με “-” τότε η μεταβλητή γίνεται on και μόνο τότε το πρόγραμμα μπορεί να αρχίσει αποθηκεύει την κάθε γραμμή του αρχείου. Αυτό γίνεται για να αποτραπούν σφάλματα στα οποία το αρχείο έχει στις πρώτες γραμμές του οτιδήποτε άλλο παρά “-”. Αφού σιγουρευτήκαμε ότι το αρχείο αρχίζει με γραμμή που υπάρχει μόνο ο χαρακτήρας “-” βάζουμε στη λίστα to_be_checked_receipt[] όλες τις γραμμές μέχρι και την επόμενη γραμμή που έχει μόνο τον χαρακτήρα “-”. Όταν ολοκληρωθεί η διαδικασία έχουμε μια λίστα της μορφής

```
----------------------------------------
ΑΦΜ: 1598713974
ΤΖΑΤΖΙΚΙ: 2  1.94  3.88
ΣΥΝΟΛΟ:   3.88
----------------------------------------

ή

----------------------------------------
δφγδφγ
δασφ
----------------------------------------
```

όπου κάθε γραμμή της απόδειξης αντιστοιχεί και σε ένα κελί της λίστας. Ο μόνος περιορισμός που έχει μια απόδειξη για να αποθηκευτεί στην λίστα είναι το να αρχίζει και να τελειώνει με γραμμή που περιέχει μόνο τον χαρακτήρα “-”. Τέλος αφού έχουμε την λίστα to_be_checked_receipt[] με μία ολοκληρωμένη απόδειξη τότε την περνάμε στη συνάρτηση validate_receipt() οπού γίνεται έλεγχος ορθότητας και αν τον περάσει τότε αποθηκεύεται στο βιβλίο.

B) **Πώς ελέγχετε αν η απόδειξη είναι σωστή;**
Το ερώτημα απαντάται στο δεύτερο μέρος (διαδικασίας επίλυσης) όταν αναλύεται η συνάρτηση validate_receipt()

C) **Τι δομές δεδομένων χρησιμοποιείτε για να απαντάτε γρήγορα τα ερωτήματα του χρήστη και γιατί τις επιλέξατε;**
Η δομή δεδομένων που χρησιμοποιήσαμε για να απαντήσουμε τα ερωτήματα του χρήση όσο το δυνατό γρηγορότερα φαίνεται εκτενώς στο δεύτερο μέρος (διαδικασίας επίλυσης) όταν αναλύεται η συνάρτηση fill_the_structure(). Ο λόγος που επιλέξαμε να χρησιμοποιήσουμε λεξικά κατά την αποθήκευση των αποδείξεων και των επιμέρους προϊόντων τους είναι γιατί προσφέρουν γρηγορότερη εύρεση μέχρι και 100,000 φορές φυσικά χρησιμοποιώντας λεξικά χρειαζόμαστε περισσότερη μνήμη σε σχέση με τις λίστες.



* Η εικόνα όπως και τα νούμερα που αναφέρονται στην απάντηση είναι ιδιοκτησία της ιστοσελίδας “https://www.jessicayung.com/python-lists-vs-dictionaries-the-space-time-tradeoff/” και η αναφορά σε αυτά κάνει χρήση του fair Use.

D) **Πώς εξασφαλίζετε ότι μπορείτε να διαβάσετε πάρα πολύ μεγάλα αρχεία;**
Ο τρόπος που διαβάζονται τα αρχεία είναι με την χρήση δείκτη στο αρχείο. Δηλαδή δεν φορτώνουμε το κάθε αρχείο στην μνήμη για επεξεργασία αλλά την τοποθεσία του στον δίσκο. Έπειτα με τη χρήση του “for line in fp” φορτώνουμε στην μνήμη κάθε φορά μόνο την γραμμή του αρχείου την οποία θέλουμε να επεξεργαστούμε. Όταν το for – loop κάνει επανάληψη η προηγούμενη γραμμή σβήνεται από την μνήμη εκτός και αν την αποθηκεύσαμε σε κάποια μεταβλητή και φορτώνεται στη μνήμη η καινούρια γραμμή. Με αυτό το τρόπο δεν υπάρχουν προβλήματα μνήμης για μεγάλα αρχεία στο πρόγραμμα αλλά υπάρχουν καθυστερήσεις στο διάβασμα λόγο του ότι κάθε φορά κοιτούμε στον δίσκο και όχι στην μνήμη.

4) Υποσημείωση:

Το πρότζεκτ όπως και η αναφορά υλοποιήθηκαν από κοινού και ευθύνη για την περάτωση των επιμέρους κομματιών του φέρουμε αμφότεροι στον ίδιο βαθμό.