Εργαλεία ανάπτυξης λογισμικού και προγραμματισμός συστημάτων

Αναφορά 2ης άσκησης

Μουδήρη Κωνσταντίνα 2012030128

Μπαρτς Μάνουελ Ράφαελ 2013030098

Για την δεύτερη άσκηση του μαθήματος μας ζητήθηκε να υλοποιήσουμε ένα πρόγραμμα computeSales.py σε Python που διαβάζει αρχεία με αποδείξεις πώλησης προϊόντων, κάνει έλεγχο ορθότητας και εκτυπώνει στατιστικά στοιχεία ανάλογα με το τι επιλέγει ο χρήστης.

Όταν ο χρήστης πατήσει 1, δίνει ένα αρχείο και εάν είναι έγκυρο το διαβάζουμε γραμμή-γραμμή. Έχουμε μία μεταβλητή "case" που μας δείχνει τι θα έπρεπε να περιέχει η γραμμή στην οποία βρισκόμαστε. Ξεκινάμε ελέγχοντας ότι η πρώτη γραμμή περιέχει μόνο τον χαρακτήρα '-' (case = 0) και θέτουμε case = 1 για την επόμενη γραμμή. Στην συνέχεια ελέγχουμε ότι η γραμμή αυτή θα είναι της μορφής ΑΦΜ: ΑΑΑΑΑΑΑΑΑ με Α να αντιστοιχεί σε ένα ψηφίο και το case γίνεται ίσο με 2.

Στην επόμενη γραμμή που θα διαβάσουμε περιμένουμε να υπάρχει το πρώτο προϊόν της απόδειξης, ελέγχουμε την ορθότητα της σύνταξης και αν η ποσότητα, η τιμή μονάδας και η συνολική τιμή είναι αριθμοί (ελέγχουμε με try, except). Αν είναι όντως αριθμοί τότε ελέγχουμε αν για το συγκεκριμένο προϊόν η ποσότητα επί την τιμή της μονάδας ισούται με την συνολική τιμή και θέτουμε case = 3.

Έτσι στην επόμενη γραμμή περιμένουμε είτε ακόμα ένα προϊόν ή το σύνολο της απόδειξης. Αν κάνοντας split στην γραμμή πάρουμε 2 στοιχεία τότε αναμένουμε γραμμή που θα περιέχει σύνολο και ελέγχουμε αν είναι της μορφής ΣΥΝΟΛΟ: ΣΥΝΟΛΙΚΟ_ΠΟΣΟ. Το ΣΥΝΟΛΙΚΟ_ΠΟΣΟ πρέπει να είναι αριθμός (το ελέγχουμε με try, except) και αν δεν έχουμε σφάλμα σε προηγούμενη γραμμή της απόδειξης ελέγχουμε αν ο αριθμός αυτός είναι ίσος με το άθροισμα των συνολικών τιμών όλων των προϊόντων. Αλλιώς εάν κάνοντας split πάρουμε 4 στοιχεία αναμένουμε κι άλλο προϊόν στην απόδειξη και κάνουμε τους ίδιους ελέγχους με την περίπτωση case = 2 (όπου αναμένουμε το πρώτο προϊόν της απόδειξης). Όταν συναντήσουμε γραμμή που περιμένουμε να δούμε ΣΥΝΟΛΟ αλλάζουμε το case ώστε να γίνει και πάλι 0 για να ελέγξουμε ότι η επόμενη γραμμή περιέχει μόνο τον χαρακτήρα '-'.

Σε κάθε νέα γραμμή που διαβάζουμε ελέγχουμε αν βρισκόμαστε σε γραμμή με παύλες '-'. Εάν βρισκόμαστε στην πρώτη γραμμή που έχει μόνο παύλες, δηλαδή στην αρχή της πρώτης απόδειξης του αρχείου, δεν αποθηκεύουμε τίποτα. Διαφορετικά είμαστε στο τέλος μιας απόδειξης. Ελέγχουμε ότι έχουμε περάσει από γραμμή όπου αναμέναμε σύνολο και ότι δεν υπήρχε κανένα σφάλμα, δηλαδή η απόδειξη ήταν σωστή, για να αποθηκεύσουμε τα στοιχεία της απόδειξης που προηγήθηκε.

Έπειτα έχοντας κάνει όλους τους απαραίτητους ελέγχους σε κάθε γραμμή χωρίς να έχουμε σφάλμα και ενώ βρισκόμαστε σε γραμμή που περιέχει προϊόν αποθηκεύουμε σε προσωρινά λεξικά ό,τι μας χρειαστεί στην συνέχεια. Συγκεκριμένα για την διευκόλυνση της υλοποίησης του δεύτερου ερωτήματος έχουμε δημιουργήσει ένα λεξικό (tmp_d2) που περιέχει ζεύγη με

όνομα προϊόντος (κλειδί) και ένα άλλο λεξικό (τιμή), το οποίο περιέχει ζεύγη με ΑΦΜ και συνολικό ποσό πωλήσεων (λεξικό τύπου 1). Αν στο tmp_d2 υπάρχει ήδη το προϊόν της γραμμής στην οποία βρισκόμαστε(πγ) τότε παίρνουμε το λεξικό τύπου 1 που αντιστοιχεί σ' αυτό το προϊόν και το εκχωρούμε στη μεταβλητή tmp_d. Διαφορετικά το tmp_d θα είναι ένα κενό λεξικό. Ενημερώνουμε κατάλληλα το λεξικό tmp_d με βάση τα στοιχεία της γραμμής που βρισκόμαστε. Τέλος, ενημερώνουμε το tmp_d2 ώστε για το συγκεκριμένο προϊόν(πγ) το λεξικό που του αντιστοιχεί πλέον είναι το tmp_d (tmp_d2[πγ]=deepcopy(tmp_d)). Επίσης για την διευκόλυνση της υλοποίησης του τρίτου ερωτήματος χρησιμοποιήσαμε αντίστοιχη λογική με το πρώτο λεξικό(tmp_d3) να περιέχει ζεύγη με ΑΦΜ και δεύτερο λεξικό, το οποίο τώρα θα έχει ζεύγη με όνομα προϊόντος και συνολικό ποσό. Παρακάτω βλέπουμε τον αντίστοιχο κώδικα.

```
if mistake == 0:
    x = l.split(":")
    x[0] = x[0].upper()     # onoma proiontos tis grammis stin opoia vriskomaste
    if x[0] == "ΑΦΜ":
        j = x[1].split()
        afm = j[0]          # AFM tis apodeiksis stin opoia vriskomaste

    if x[0] != "ΑΦΜ" and x[0] != "ΣΥΝΟΛΟ" and len(x)>1:
        if x[0] in tmp_d2:
            tmp_d=tmp_d2[x[0]]
        else:
            tmp_d={}
        if afm in tmp_d3:
            tmp_3=tmp_d3[afm]
        else:
            tmp_3={}
        stoixeia = x[1].split()
        total_price = stoixeia[2]
```

```
        if afm in tmp_d:
            # print (count, float(tmp_d[afm]), float(total_price) )
            tmp_d[afm] = round(float(tmp_d[afm]) + float(total_price),2)
        else:
            tmp_d[afm] = total_price
        # print(total_price, float(total_price))

        if x[0] in tmp_3:
            # print (count, float(tmp_d[afm]), float(total_price) )
            tmp_3[x[0]] = round(float(tmp_3[x[0]]) + float(total_price),2)
        else:
            tmp_3[x[0]] = total_price

        tmp_d2[x[0]]=deepcopy(tmp_d)
        tmp_d3[afm]=deepcopy(tmp_3)
l = f.readline()
```

Κάθε φορά που βρισκόμαστε στο τέλος μίας απόδειξης (γραμμή με παύλες '-'), εάν η απόδειξη δεν είχε λάθος τότε αποθηκεύουμε στα μόνιμα λεξικά(monimo_d2, monimo_d3) τις τιμές των προσωρινών λεξικών (tmp_d2, tmp_d3). Αντίθετα εάν στην απόδειξη που προηγήθηκε εντοπίστηκε κάποιο λάθος τότε αυτό σημαίνει ότι όλες οι αλλαγές που ενδεχομένως έχουν γίνει στα προσωρινά λεξικά δεν πρέπει να τις λάβουμε υπόψιν, οπότε αποθηκεύουμε σε αυτά τις τιμές που έχουμε στα μόνιμα λεξικά. Με αυτόν τον τρόπο εξασφαλίζουμε ότι οι αλλαγές που έχουν γίνει σε μία απόδειξη που τελικά ήταν λάθος, θα αναιρεθούν.

Όταν ο χρήστης πατήσει 2, του ζητάμε να δώσει ένα όνομα προϊόντος και στην συνέχεια το ψάχνουμε στο λεξικό monimo_d2 το οποίο περιέχει ζεύγη με όνομα προϊόντος (κλειδί) και ένα άλλο λεξικό (τιμή). Αυτό περιέχει ζεύγη με ΑΦΜ(κλειδί) και συνολικό ποσό πωλήσεων (λεξικό τύπου 1). Έτσι αν το προϊόν που έδωσε ο χρήστης υπάρχει στο monimo_d2 τότε παίρνουμε το αντίστοιχο λεξικό τύπου 1 και το ταξινομούμε με αύξουσα σειρά με βάση το ΑΦΜ πριν το εκτυπώσουμε. Διαφορετικά δεν εκτυπώνουμε τίποτα.

Όταν ο χρήστης πατήσει 3, του ζητάμε να δώσει ένα ΑΦΜ και στην συνέχεια το ψάχνουμε στο λεξικό monimo_d3 το οποίο περιέχει ζεύγη με ΑΦΜ(κλειδί) και δεύτερο λεξικό(λεξικό τύπου 2). Αυτό έχει όνομα προϊόντος(κλειδί) και συνολικό ποσό(τιμή). Έτσι αν το ΑΦΜ που έδωσε ο χρήστης υπάρχει στο monimo_d3 τότε παίρνουμε το αντίστοιχο λεξικό τύπου 2 και το ταξινομούμε με αύξουσα σειρά με βάση το όνομα προϊόντος πριν το εκτυπώσουμε. Διαφορετικά δεν εκτυπώνουμε τίποτα.

Τέλος αν ο χρήστης πατήσει 4, πραγματοποιούμε έξοδο από το πρόγραμμα.

Για την υλοποίηση του προγράμματός μας χρησιμοποιήσαμε λεξικά καθώς όταν διαβάζουμε ένα αρχείο λόγω των ερωτημάτων 2 και 3, μας συμφέρει να αποθηκεύουμε τις πληροφορίες σε ζεύγη προϊόν-λεξικό και ΑΦΜ-λεξικό αντίστοιχα. Έτσι για ένα συγκεκριμένο προϊόν ή ΑΦΜ είναι εύκολο να πάρουμε το αντίστοιχο λεξικό το οποίο περιέχει τα ζεύγη που πρέπει να εκτυπωθούν σε κάθε ερώτημα.

Αξίζει να σημειωθεί ότι ο κώδικάς μας λειτουργεί σωστά ακόμα και όταν δοθεί αρχείο που περιέχει ονόματα προϊόντων που αποτελούνται από δύο ή περισσότερες λέξεις χωρισμένες με κενό.

Τέλος, παραθέτουμε κάποια παραδείγματα εκτέλεσης του προγράμματος:

```
C:\Users\user\Python>python project2final2.py
Give your preference: (1: read new input file, 2: print statistics for a specific product, 3: print statistics for a specific AFM, 4: exit the program)1
Give new file
receiptFileBIG.txt
Give your preference: (1: read new input file, 2: print statistics for a specific product, 3: print statistics for a specific AFM, 4: exit the program)2
Give product name
τζατζικι
1133528586 46039.3
1375031773 44465.44
1494754516 44388.18
1667054375 44154.7
2069228931 41599.34
2139549311 45792.5
2196911440 48335.36
2725955513 42778.76
2899293871 43699.7
Give your preference: (1: read new input file, 2: print statistics for a specific product, 3: print statistics for a specific AFM, 4: exit the program)3
Give AFM
1133528586
AMSTEL 4457.14
ΚΡΑΣΙ ΡΟΖΕ 3983.66
ΜΠΡΙΖΟΛΑ ΜΟΣΧΑΡΙΣΙΑ 16870.78
ΜΠΡΙΖΟΛΑ ΧΟΙΡΙΝΗ 5265.86
ΠΟΙΚΙΛΙΑ 45205.72
ΤΖΑΤΖΙΚΙ 46039.3
ΦΙΛΕΤΟ ΚΟΤΟΠΟΥΛΟ 5140.14
Give your preference: (1: read new input file, 2: print statistics for a specific product, 3: print statistics for a specific AFM, 4: exit the program)
```

```
C:\Users\user\Python>python project2final2.py
Give your preference: (1: read new input file, 2: print statistics for a specific product, 3: print statistics for a specific AFM, 4: exit the program)1
Give new file
receiptFile.txt
Give your preference: (1: read new input file, 2: print statistics for a specific product, 3: print statistics for a specific AFM, 4: exit the program)1
Give new file
file1.txt
Give your preference: (1: read new input file, 2: print statistics for a specific product, 3: print statistics for a specific AFM, 4: exit the program)2
Give product name
φιλετο κοτοπουλο
1502228322 117.12
1598713974 237.12
2230918454 32.28
2567870682 81.12
2631610540 44.32
2834112420 281.0
2996927915 260.54
3106124650 88.46
7370682019 140.98
Give your preference: (1: read new input file, 2: print statistics for a specific product, 3: print statistics for a specific AFM, 4: exit the program)
```