1ο Project 2023-2024, Αναφορά Ομάδας

Παντελής Φλουρής, up1093507, up1093507@ac.upatras.gr

Αγγελος Μενεγάτος, up1093426, up1093426@ac.upatras.gr

Χρυσάφης Κολτσάκης, up1084671, up1084671@ac.upatras.gr

Γιώργος Αμαξοπουλος, up1093311,up1093311@ac.upatras.gr

Λειτουργουν ολα τα υποερωτηματα.

Ερώτημα 1: Shell Scripting

Αρχικοποιω το file_path με το αρχειο Businesses.csv, ωστε αμα δεν γινει επιλογη απο τον user για αλλαγη αρχειου να χρησημοποιηθει αυτο.

Μετα βαζω while true ωστε μετα απο καθε εντολη να μην φυγουμε απο το προγραμμα,

echo για να δωσουμε στον user τι να επιλεξει και μετα χρησημοποιω case.

1) αλλαζει το file_path και χρησημοποιει το file που βαλαμε, αμα δεν υπαρχει το κανει print στον user, και πρεπει να το ξαναβαλει.

2) χρηση του grep για να κανει print τα στοιχεια τις επιχειρησης με το id που του δωσαμε.

3) ο χρηστης εισαγει το id της επιχειρησης,

το προγραμμα κανει check αμα υπαρχει, και μετα βαζουμε αλλο ενα case ωστε αναλογα με

το input του user να αλλαζει και οτι του εισαγουμε. Αυτο γινεται με την χρηση της εντολης awk.

Η χρηση του 2ου case γινεται διοτι δεν καταφεραμε να κανουμε να το κανουμε να λειτουργήσει με μια awk.

4) χρηση του more λογω ερωτηματος.

5) κανει create new file και κανει τις αλλαγες

6) exit

UI

```
[1] Επιλογή αρχείου επιχειρήσεων
[2] Προβολή στοιχείων επιχείρησης
[3] Αλλαγή στοιχείου επιχείρησης
[4] Προβολή αρχείου
[5] Αποθήκευση αρχείου
[6] Έξοδος
```

[1]

```
Επιλέξτε αρχείο επιχειρήσεων
Εισάγετε το path του αρχείου. (enter για Businesses.csv):
```

```
Εισάγετε το path του αρχείου. (enter για Businesses.csv): john.doe
Το αρχείο john.doe δεν υπάρχει
```

```
Εισάγετε το path του αρχείου. (enter για Businesses.csv): asd.csv
Το επιλεγμένο αρχείο είναι: asd.csv
```

[2]

```
2
Επιλέξτε επιχείρηση
360857
360857,1478 Social Club,11 Dove Street,Glasgow,G53 7BS,-4.362002,55.812693
```

[3]

```
3
Επιλέξτε επιχείρηση
360857
360857,1478 Social Club,11 Dove Street,Glasgow,G53 7BS,-4.362002,55.812693
Επιλέξτε το στοιχείο που θέλετε να αλλάξετε:
1. ID
2. BusinessName
3. AddressLine2
4. AddressLine3
5. PostCode
6. Longitude
7. Latitude
```

[4]

```
ID,BusinessName,AddressLine2,AddressLine3,PostCode,Longitude,Latitude
19429,(Starbucks),390 Provan Walk,Glasgow,G34 9DL,-4.136909,55.872982
test,1478 Social Club,11 Dove Street,Glasgow,G53 7BS,-4.362002,55.812693
36363,1call Direct,116 West Regent Street,Glasgow,G2 2QD,-4.258314,55.863552
36362,2 Go,397 Dumbarton Road,Glasgow,G11 6BE,-4.308613,55.870617
507934,200 St Vincent Street,200 St Vincent Street,Glasgow,G2 5SG,-4.260957,55.86229
95689,279 Cafe Bistro,279 Dumbarton Road,Glasgow,G11 6AB,-4.303939,55.870516
110587,32nd Signal Regiment (v),21 Jardine Street,Glasgow,G20 6JU,-4.277246,55.877088
10859,7 Day Catering Ltd  -Staff Canteen,53 Coltness Street,Glasgow,G33 4JD,-4.152553,55.863938
406755,7th Heaven,15 Elmbank Gardens,Glasgow,G2 4NQ,-4.270439,55.865108
556088,916 - Diner,916 Sauchiehall Street,Glasgow,G3 7TF,-4.284479,55.865774
31114,Abercromby Cafe,168 Abercromby Street,Glasgow,G40 2RZ,-4.229148,55.852688
--More--(0%)
```

[5]

```
5
Enter the path to save the file or press enter for default: 123.sa
File saved to 123.sa
```

Και εκανε save σε νεο αρχειο με το ονομα που του βαλαμε.

[6]

```
[6] Έξοδος
6
gregthebig@DESKTOP-BT7GJ2V:~/ceid$
```

Ερώτημα 2: Διεργασίες

a) integral_mc_shm.c

Για την υλοποιηση αυτου του ερωτηματος, πηραμε τις δυο συναρτησεις τις οποιες ειχατε υλοποιησει για το sequential.

Για την κοινη μνημη δημιουργησαμε μια συναρτηση η οποια παιρνει σαν εισοδο τον αριθμο των διεργασιων, δημιουργει με mmap εναν πινακα (σε μια τυχαια θεση μνημης) με μια θεση double για καθε διεργασια και τον επιστρεφει.

Για τους υπολογισμους της καθε διεργασιας, υλοποιησαμε μια συναρτηση η οποια δεχεται σαν εισοδο τον αριθμο των πραξεων που πρεπει να κανει η καθε διεργασια, εναν δεικτη στην κοινη μνημη και την θεση του πινακα που πρεπει να αποθηκευσει το αποτελεσμα.

Για την τυχαια ακολουθια απο την καθε διεργασια χρησιμοποιησαμε τις συναρτησεις srand48 και drand48, που, αν και δεν ειναι πραγματικα τυχαιες, παραγουν διαφορετικες τιμες για καθε διεργασια λογω του argument στην srand48.

Μετα την παραγωγη της ψευδοτυχαιας ακολουθιας, αρχιζει μια επαναληψη μεχρι να ολοκληρωθουν οι πραξεις που πρεπει να κανει η διεργασια, στην οποια υπολογιζονται νεες τιμες μεσω της συναρτησης f και αθροιζονται σε μια μεταβλητη.

Οταν τελειωσει ο βρογχος, αποθηκευεται το συνολικο αποτελεσμα των πραξεων στην θεση του πινακα που αντιστοιχει σε αυτη την διεργασια.

Για τον υπολογισμο του τελικου αποτελεσματος, δημιουργουμε μια συναρτηση η οποια δεχεται σαν arguments τον αριθμο των διεργασιων που εκαναν υπολογισμους, εναν δεικτη στην κοινη μνημη και τον συνολικο αριθμο των πραξεων που εγιναν απο ολες τις διεργασιες. Αρχικοποιει την σταθερα που απαιτει το Monte Carlo integration, και μεσα σε ενα for loop υπολογιζει το αποτελεσμα πολλαπλασιαζοντας ολες τις θεσεις μνημης του πινακα με την σταθερα.

Επειτα επιστρεφει το αποτελεσμα.

Στην main, μετα απο την αρχικοποιηση των απαραιτητων μεταβλητων, ελεγχουμε εαν ο χρηστης εξεφρασε ποσες διεργασιες θελει να υπολογισουν.

Εαν επιλεξε, αποθηκευουμε τον αριθμο στην μεταβλητη Processes, αλλιως κραταμε την αρχικοποιημενη τιμη των 4 διεργασιων.

Υπολογιζουμε τον αριθμο των πραξεων που θα κανει η καθε διεργασια διαιρωντας τον αριθμο των πραξεων που πρεπει να γινουν με τον αριθμο των διεργασιων, και υλοποιουμε την κοινη μνημη καλώντας την συναρτηση που εχουμε φτιαξει.

Διαβαζουμε μια χρονικη στιγμη (για το τελικο αποτελεσμα) και μπαινουμε σε ενα for loop, που θα επαναληφθει μια φορα για καθε διεργασια που εχει επιλεγει να δημιουργηθει.

Στο for, καλουμε την fork() και δημιουργουμε μια διαδικασια-παιδι. Η διαδικασια παιδι εκτελει απευθειας την συναρτηση calculate_integral (για τους υπολογισμους της), και μολις επιστρεψει η συναρτηση τελειωνει.

Εν τω μεταξυ εχουμε “εκλωβισει” την διεργασια γονεα σε μια κατασταση αναμονης μεχρι να τελειωσουν ολες οι διεργασιες-παιδια.

Οταν αυτο γινει, καλειται η συναρτηση combine_results, η οποια υπολογιζει το τελικο αποτελεσμα, και το ρολοι σταματά.

Τα αποτελεσματα εκτυπώνονται οπως και στο προγραμμα που μας δωθηκε.

Στο τελος του προγραμματος, ελευθερωνεται η μνημη που δημιουργησαμε.

b) integral_mc_shm_sem.c

Για την υλοποιηση αυτου του ερωτηματος, χρειαστηκε μια μικρη αλλαγη στον προηγουμενο κωδικα.

Αντι η συναρτηση κοινης μνημης να δημιουργει εναν πινακα double, δημιουργει μονο μια θεση.

Προκειμενου να αποφυγουμε race conditions και να επιβεβαιωσουμε οτι τα αποτελεσματα μας θα ειναι σωστα, χρειαστηκε να χρησιμοποιησουμε εναν σημαφορο στην συναρτηση calculate_integral. Ο σημαφορος ειναι binary, επιτρεποντας μονο μια διεργασια να κανει access την θεση μνημης την φορα, για να βεβαιωθουμε οτι δεν θα χασουμε καποιο αποτελεσμα.

Προφανως, καθως δεν εχουμε πινακα μα μονο μια τιμη, η συναρτηση που συνδυαζει τα αποτελεσματα δεν χρειαζεται loop πλεον.

Η μονη αλλη αλλαγη ειναι πως επειδη δεν ανατειθεται ποτε μια τιμη στην θεση μνημης, αλλα προσθετονται πανω στην ηδη υπαρχουσα, χρειαστηκε να αρχικοποιησουμε την θεση με τιμη 0.

Οι διεργασιες λειτουργουν ως εξης:

Δημιουργειται μια διεργασια και ξεκιναει να εκτελει τις πραξεις που της αναλογουν.

Οταν τελειωσει, ελεγει εαν καποια διεργασια χρησιμοποιει την θεση μνημης.

Εαν καποια αλλη διεργασια χρησιμοποιει την μνημη, τοτε περιμενει να τελειωσει την προσπελαση και να ελευθερωσει τον σημαφορο.

Εαν δεν την χρησιμοποιει καποια αλλη διεργασια, την καταλαμβανει και ο σημαφορος αποτρεπει την εισοδο καποιας αλλης διεργασιας.

```
gcc -Wall -O3 -o bin/integral_mc_seq integral_mc_seq.c -lm
gcc -Wall -O3 -o bin/integral_mc_shm integral_mc_shm.c -lm
gcc -Wall -O3 -o bin/integral_mc_shm_sem integral_mc_shm_sem.c -lm
----------------------------------------
Starting ./bin/integral_mc_seq
Starting ./bin/integral_mc_shm 12
Starting ./bin/integral_mc_shm_sem 12
----------------------------------------
Total elapsed time: 16 seconds
----------------------------------------
Results of ./bin/integral_mc_seq:
Result=0.7386452092778891 Error=2.211241e-06 Rel.Error=2.993653e-06 Time=14.384724 seconds

Results of ./bin/integral_mc_shm 12:
Result=0.7386395441884999 Error=3.453848e-06 Rel.Error=4.675937e-06 Time=5.942066 seconds

Results of ./bin/integral_mc_shm_sem 12:
Result=0.7386395441884999 Error=3.453848e-06 Rel.Error=4.675937e-06 Time=5.929361 seconds

rm -f bin/*
```

Ετσι εχουμε σαν αποτελεσμα να χρειαζομαστε λιγοτερη μνημη για το ιδιο αποτελεσμα, σε πιο γρηγορο χρονο(!!), καθως δεν χρειαζομαστε το loop για την συνδεση των αποτελεσματων.

Ερώτημα 3: Συγχρονισμός Διεργασιών

Αρχικά, αφού διαβασαμε την εκφώνηση, σκεφτηκαμε ότι ο ψευδοκώδικας μας θα αποτελείται από δυο μέρη. Ένα μέρος που θα βαλουμε όλες τις κοινές μεταβλητές και τους σημαφόρους μας, και ένα που θα περιέχει τον κώδικα για τις τρείς διεργασίες, τις take_food(), cook food(), και function helper().

Η σκέψη μας για την επιλογή των μεταβλητών και των σημαφόρων ήταν η εξής.
Χρειάζεται κάπως να μετράμε πόσες μερίδες φαγητό υπάρχουν μέσα στην κατσαρόλα. Έτσι όρισα μια ακέραιη μεταβλητή, την pot_portions, που θα μετράει ακριβώς αυτό. Αντιστοίχως Δημιουργησαμε και τις μεταβλητές beans και carrots. Επίσης θέλουμε έναν σημαφόρο για όποτε μπαίνουμε στην κρίσιμη περιοχή του κώδικα, δηλαδή όποτε πάμε να αλλάξουμε τιμές στις μεταβλητές, τον οποίο ονομαζουμε lock και αρχικοποιώ στο 1. Τέλος φτιαξαμε από έναν σημαφόρο για τον μάγειρα και τους δύο βοηθούς.

Στη συνέχεια ξεκινησαμε την υλοποίηση των διεργασιών:
Για την take_food, υπάρχουν δύο καταστάσεις. Ή θα υπάρχει διαθέσιμη μερίδα φαγητό , οπότε κάποιος θα σερβιριστεί και οι μερίδες θα ελαττωθούν κατά μία, είτε δεν θα υπάρχουν και έτσι θα πρέπει να περιμένει τον μάγειρα να μαγειρέψει. Όσο ο σημαφόρος cook_semaphore = 0 , σημαίνει ότι ο μάγειρας μαγειρεύει. Όταν μαγειρέψει, ο σημαφόρος θα ενημερωθεί στην τιμή 1(αυτό γίνεται σε πιο κάτω μέρος του κώδικα). Όταν ξανά τελειώσει το φαγητό θα τρέξει το εάν (pot_portions == 0), αλλά τώρα δεν θα υπάρχει φαγητό και ο cook_semaphore θα είναι 1. Οπότε στο wait(cook_semaphore) δεν θα γίνει ενεργός αναμονή , το cook_semaphore θα γίνει 0, το πρόγραμμα θα ξαναμπεί στο εάν (pot_portions == 0), αλλά τώρα θα περιμένει όπως πρέπει στο wait. Κάτι αντίστοιχο θα συμβαίνει και στις cook_food kai helper.

Για την cook_food, εάν υπάρχουν διαθέσιμα υλικά, τότε αυτά θα χρησιμοποιηθούν και οι

μερίδες θα αυξηθούν. Αλλιώς, εάν δεν υπάρχουν τα υλικά θα περιμένει μέχρι να τα φέρουν οι βοηθοί.

Για τη helper, εάν λείπουν φασόλια, τότε ο ένας βοηθός θα πάει αν φέρει. Το γεγονός ότι οι βοηθοί μπορούν να μεταφέρουν ορισμένη ποσότητα υλικών προγραμματίζεται απλά με έναν βρόγχο while(δεν φτάνουν τα υλικά), κάθε βρόγχος αντιστοιχεί σε μια διαδρομή πήγαινε έλα ενός βοηθού. Όταν φέρει όσα φασόλια χρειάζονται τότε ειδοποιεί με το signal ότι έφερε τα υλικά. Ομοίως και για τα καρότα.

Σε γενικές γραμμές αυτό το ερώτημα μας φάνηκε αρκετά εύκολο και δεν αντιμετωπισαμε κάποια δυσκολία. Όμως επειδή η απάντηση είναι σε ψευδοκώδικα, δεν μπορουμε να τρεξουμε τον κώδικά και ειμαστε σιγουροι.

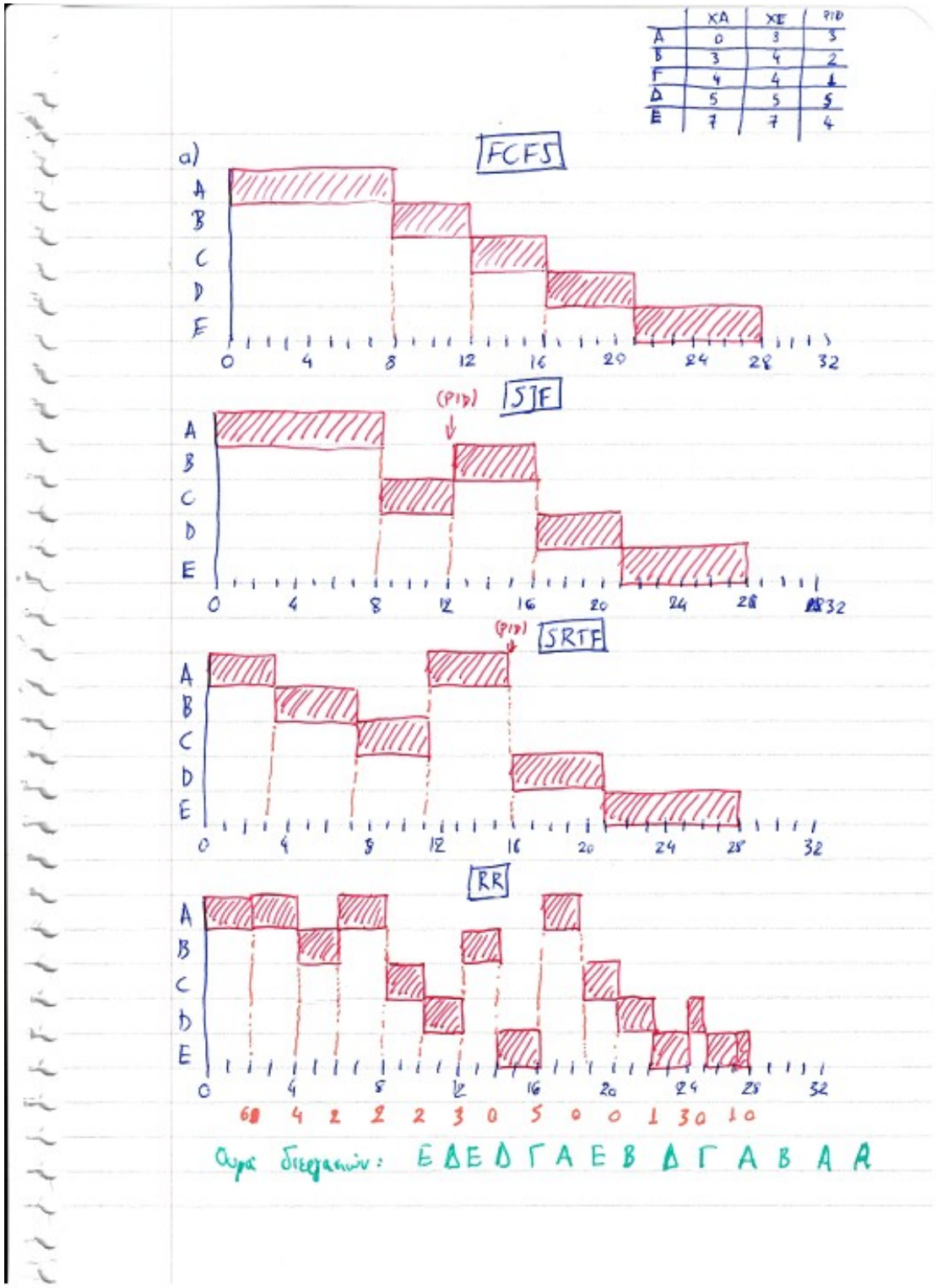
```
shared variables:

int pot_portions = 0
int beans = 0
int carots = 0
lock = Semaphore(1)
cook_semaphore = Semaphore(0)                    //me 0 o mageiras mageireuei.
ass1_semaphore = Semaphore(0)                    //me 0 oi boithoi metaferoun ulika, ass1=beans
ass2_semaphore = Semaphore(0)                    //ass2=carots



function take_food():
        while(true)
        {
                if (pot_portions > 0)
                {
                        wait(lock)                                      //an kapoios allos serbiretai perimene, allios serbire esu
                        pot_portions --                                 //serbirei faghto ston eayto toy
                        signal(lock)                            //telos serbirismatos, mporei na serbirei allos twra
                }
```

```
                else if (pot_portions == 0)
                {
                        wait(cook_semaphore)        //perimenei na teleiosei o mageiras to mageirema.
                }
        }

function cook food():
        while(true)
        {
                if(beans >= ΣΥ+3 AND carots >= ΣΥ)
                {
                        wait(lock)
                        beans = beans - ΣΥ+3
                        carots = carots - ΣΥ
                        pot_portions ++
                        signal(lock)
                        signal(cook_semaphore)      // o mageiras mageirepse, cook_semaphore=1
                }
                else
                {
                        wait(ass1_semaphore)
                        wait(ass2_semaphore)        //o mageiras perimenoi tous bohthoys na teleiosoun me thn metafora
                }
        }

function helper():
{
        while(true)
        {
                if(beans<ΣΥ+3)
                {
                        while(beans<ΣΥ+3)
                        {
                                wait(lock)
                                beans = beans + ΣΥ
                                signal(lock)
                        }
                        signal(ass1_semaphore)      //o boithos gia ta beans efere ta ulika
                }
```

```
                    if(carots<ΣΥ)
                    {
                              while(carots<ΣΥ)
                              {
                                        wait(lock)
                                        carots = carots + ΣΥ
                                        signal(lock)
                              }
                              signal(ass2_semaphore)        //o boithos gia ta carots efere ta ulika
                    }
          }
}
```

Ερώτημα 4: Χρονοπρογραμματισμός Διεργασιών

α)



Β)

Διεργασία	Χρόνος Αναμονής	Χρόνος Απόκρισης	Χρόνος Ολοκλήρωσης	#Θεματικές Αλλαγές
Α	0-0=0	0-0=0	8-0=8	1
Β	8-3=5	8-3=5	12-3=9	1
Γ	12-4=8	12-4=8	16-4=12	1
Δ	16-5=11	16-5=11	21-5=16	1
Ε	21-7=14	21-7=14	28-7=21	1
Μέσος Όρος	38/5=7.6	38/5=7.6	66/5=13.2	1

FCFS

Διεργασία	Χρόνος Αναμονής	Χρόνος Απόκρισης	Χρόνος Ολοκλήρωσης	#Θεματικές Αλλαγές
Α	0	0	8	1
Β	9	9	13	1
Γ	4	4	8	1
Δ	11	11	16	1
Ε	14	14	21	1
Μέσος Όρος	38/5=7.6	38/5=7.6	66/5=13.2	1

SJF

Διεργασία	Χρόνος Αναμονής	Χρόνος Απόκρισης	Χρόνος Ολοκλήρωσης	#Θεματικές Αλλαγές
Α	8	0	16	2
Β	0	0	4	1
Γ	3	3	7	1
Δ	11	11	16	1
Ε	14	14	21	1
Μέσος Όρος	36/5=7.2	28/5=5.6	64/5=12.8	1.2

SRTF

Διεργασία	Χρόνος Αναμονής	Χρόνος Απόκρισης	Χρόνος Ολοκλήρωσης	#Θεματικές Αλλαγές
Α	10	0	19	3
Β	7	1	11	2
Γ	12	4	16	2
Δ	13	5	20	3
Ε	14	7	21	3
Μέσος Όρος	56/5=11.2	17/5=3.4	87/5=17.4	13/5=2.6

RR

Γ) Έχουμε νέο ΜΧΑ 7,2*3 = 21,6 και 12 καθυστερήσεις λόγω θεματικών αλλαγών. Αφαιρούμε από το νέο ΜΧΑ το παλιό (21,6-11,2 = 10,4) και στη συνέχεια διαιρούμε τη συνολική καθυστέρηση με τον αριθμό των διαφορετικών καθυστερήσεων. Παίρνουμε 10,4/12 = 0,87 μονάδες χρόνου για κάθε θεματική αλλαγή. Παρατηρούμε επίσης ότι το νέο διάγραμμα έχει ίδια σειρά διεργασιών με το προηγούμενο από το (α).