ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής

Λειτουργικά Συστήματα
Ακαδημαϊκό Έτος 2018-2019

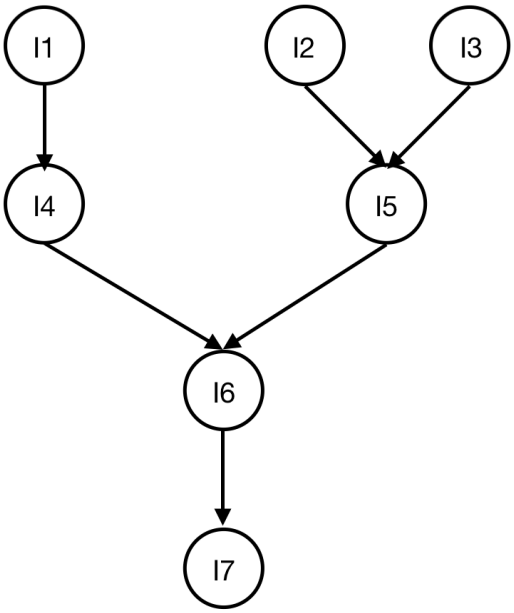
1η Εργαστηριακή Άσκηση

Ονοματεπώνυμο:	**ΑΜ:**
Αγγελής Πέτρος:	**236268**
Δημητρούκα Γιαννούλα:	**236291**

ΜΕΡΟΣ 2

ΕΡΩΤΗΜΑ Α

i)



ii)

```
COBEGIN
        BEGIN
        I1;
        I4;
        END;
        BEGIN
                COBEGIN
                I2;
                I3;
                COEND;
        I5;
        END;
I6;
I7;
```

iii)

```
  Y, X, E, F, G, H, I, Y11, Y12, C, D, Y13, Y21, Y22, Y31 integer;
  S1=S2=S3=S4=S5=S6= 0;
COBEGIN
  I1 : BEGIN Y11 : = C + D ; SIGNAL(S1) ; END;
  I2 : BEGIN Y12 : = E - F ; SIGNAL(S2) ;  END;
  I3 : BEGIN Y13 : = H + I ; SIGNAL(S3) ; END;
  I4 : BEGIN WAIT(S1) ; Y21 : = X * Y11 ; SIGNAL(S4) ;  END;
  I5 : BEGIN WAIT(S2) ; WAIT(S3) ; Y22 : = Y12 * Y13; SIGNAL(S5) ; END;
  I6 : BEGIN WAIT(S4) ; WAIT(S5) ; Y31 : = Y21 / Y22 ; SIGNAL(S6) ; END;
  I7 : BEGIN WAIT(S6) ; Y = G + X31; END;
COEND;
```

ΕΡΩΤΗΜΑ Β

i)

shared int x, y;	
Διεργασία Δ1	**Διεργασία Δ2**

A1: x = 1;
A2: y = 2;
if (x == y)
  print "X";
	B1: x = 2;
	B2: y = 2;
	if (x == y)
	  print "Y";

Περιπτώσεις:

- **A1→A2→B1→B2: Y**
- **A1→B1→A2→B2: XY**
- **A1→B1→B2→A2: YX**
- **B1→A1→B2→A2: -**
- **B1→A1→A2→B2: -**
- **B1→B2→A1→A2: Y**

ii)

<table>
<tr><td colspan="2">var x, y : shared integer;
var s1, s2 : semaphore;
begin
s1 := 0; s2 := 0;</td></tr>
<tr><td colspan="2">cobegin</td></tr>
<tr><td>Διεργασία Δ1</td><td>Διεργασία Δ2</td></tr>
<tr><td>begin
A1: x = 1;
signal(s1);
A2: y = 2;
wait(s2);
if (x == y)
  print “X”;
signal(s1);
end</td><td>begin
wait(s1);
B1: x = 2;
signal(s2);
B2: y = 2;
wait(s1);
if (x == y)
  print “Y”;
end</td></tr>
<tr><td colspan="2">coend</td></tr>
</table>

Η εντολή Α1 είναι στο κώδικα της διεργασίας Δ1, ενώ η Β1 είναι εντολή στον κώδικα της διεργασίας Δ2. Επομένως, για να εξασφαλιστεί ότι η εκτέλεση της Β1 έπεται της Α1, μετά από την Α1 στον κώδικα της διεργασίας Δ1, εισάγουμε την εντολή **signal(s1)**. Πριν την Β1, στον κώδικα της διεργασίας Δ2, εισάγουμε την εντολή **wait(s1).**
Έπειτα για να εξασφαλίσουμε ότι η εντολή Α2 της διεργασίας Δ1, εκτελείται μετά την εντολή Α1 και Β1, εισάγουμε την εντολή **signal(s2)** πριν την εντολή Β2 της διεργασίας Δ2 και την εντολή **wait(s2)** μετά την εντολή Α2 της διεργασίας Δ1.
Στη διεργασία Δ0 εισάγουμε ξανά την εντολή **signal(s1)** μετά την εντολή εκτύπωσης του «Χ». Τέλος στη διεργασία Δ1, μετά την εντολή Β2, εισάγουμε την **wait(s1)** έτσι ώστε να εξασφαλίσουμε και την εκτύπωση του «Υ» αμέσως μετά του «Χ».

Επομένως χρησιμοποιούμε δυο σημαφόρους για να εξασφαλιστεί η σειρά που θα μας οδηγήσει στο επιθυμητό αποτέλεσμα, δηλαδή το «ΧΥ». Δεν γίνεται να χρησιμοποιήσουμε μόνο ένα σημαφόρο, διότι η διεργασία Δ1 πρέπει να περιμένει την διεργασία Δ2 και αμέσως μετά η διεργασία Δ2 πρέπει να περιμένει την διεργασία Δ1. Έτσι αν χρησιμοποιήσουμε ένα μόνο σημαφόρο μπορεί να οδηγηθούμε σε αδιέξοδο ή στη παραβίαση της ορθής εκτέλεσης των εντολών.

ΕΡΩΤΗΜΑ Γ

<table>
<tr><td colspan="2">shared boolean flag[2]; /* αρχικά flag[0]=flag[1]=FALSE*/
shared turn[2]; /* αρχικά turn[0]=0 και turn[1]=0*/</td></tr>
<tr><td>Διεργασία Δ0</td><td>Διεργασία Δ1</td></tr>
<tr><td>flag[0]=TRUE;
turn[0]=(turn[1]+0)mod2;
repeat noop
until
(flag[1]==FALSE)OR(turn[0]<>(turn[1]+0
)mod2);
<ΚΡΙΣΙΜΟ ΤΜΗΜΑ>
flag[0]= FALSE;</td><td>flag[1]=TRUE;
turn[1]=(turn[0]+1)mod2;
repeat noop
until
(flag[0]==FALSE)OR(turn[1]<>(turn[0]+1
)mod2);
<ΚΡΙΣΙΜΟ ΤΜΗΜΑ>
flag[1]= FALSE;</td></tr>
</table>

Για να εξετάσουμε αυτές τις διεργασίες θα χρησιμοποιήσουμε τον αλγόριθμο του Peterson. Γενικότερα στον αλγόριθμο αυτόν, οι κοινές μεταβλητές δημιουργούνται και αρχικοποιούνται πριν την εκτέλεση των διεργασιών.
Στη συγκεκριμένη περίπτωση υπάρχουν δυο πίνακες από flags, flag[] και turn[].

1Η ΠΕΡΙΠΤΩΣΗ:

Διεργασία Δ0	**Διεργασία Δ1**

flag[0]=TRUE;
turn[0]=(turn[1]+0)mod2;
 	
-Γίνεται έλεγχος της συνθήκης που βρίσκεται στο βρόχο επανάληψης. Η συνθήκη είναι false καθώς flag[1]=TRUE και turn[0]=(turn[1]+0)mod2. Δεδομένου αυτού δεν υπάρχει αναμονή στον βρόχο.
-Εισβάλλει στο κρίσιμο τμήμα και η διαδικασία χάνει τον επεξεργαστή.	

	flag[1]=TRUE;
	turn[1]=(turn[0]+1)mod2;
	
	-Γίνεται έλεγχος της συνθήκης που βρίσκεται στο βρόχο επανάληψης.
	-Η συνθήκη είναι true, οπότε η διεργασία είναι απασχολημένη μέχρι να χάσει τον επεξεργαστή.

-Η Δ0 ξαναρχίζει και συνεχίζει μέχρι να τερματίσει στο κρίσιμο τμήμα της. -Φεύγει από το κρίσιμο τμήμα flag[1]= FALSE -Αρχίζει να εκτελείται ξανά η διεργασία, εκτός από το κρίσιμο τμήμα της.	
	-Γίνεται έλεγχος της συνθήκης που βρίσκεται στο βρόχο επανάληψης. -Η συνθήκη είναι false, καθώς flag[0] = FALSE, επομένως δεν περιμένει αλλά εισβάλλει στο κρίσιμο τμήμα.

2Η ΠΕΡΙΠΤΩΣΗ:

ΔΙΕΡΓΑΣΙΑ Δ0	**ΔΙΕΡΓΑΣΙΑ Δ1**

flag[0]=TRUE;
turn[0]=(turn[1]+0)mod2;
 	
-Εδώ η διεργασία χάνει τον επεξεργαστή.	

	flag[1]=TRUE;
	turn[1]=(turn[0]+1)mod2;
	
	-Γίνεται έλεγχος της συνθήκης που βρίσκεται στο βρόχο επανάληψης.
	-Η συνθήκη είναι true, οπότε η διεργασία είναι απασχολημένη μέχρι να χάσει τον επεξεργαστή.

-Γίνεται έλεγχος της συνθήκης που βρίσκεται στο βρόχο επανάληψης. -Η συνθήκη είναι false, γιατί Δεδομένου αυτού δεν υπάρχει αναμονή στον βρόχο. -Εισβάλλει στο κρίσιμο τμήμα	

3Η ΠΕΡΙΠΤΩΣΗ:

ΔΙΕΡΓΑΣΙΑ Δ0	**ΔΙΕΡΓΑΣΙΑ Δ1**
flag[0]=TRUE; -Εδώ η διεργασία χάνει τον επεξεργαστή.	

	flag[1]=TRUE;
	turn[1]=(turn[0]+1)mod2;
	
	-Γίνεται έλεγχος της συνθήκης που βρίσκεται στο βρόχο επανάληψης.
	-Η συνθήκη είναι true, οπότε η διεργασία είναι απασχολημένη μέχρι να χάσει τον επεξεργαστή.

turn[0]=(turn[1]+0)mod2; -Γίνεται έλεγχος της συνθήκης που βρίσκεται στο βρόχο επανάληψης. -Η συνθήκη είναι true, οπότε η διεργασία είναι απασχολημένη μέχρι να χάσει τον επεξεργαστή.	
	-Γίνεται ξανά έλεγχος της συνθήκης που βρίσκεται στο βρόχο επανάληψης. -Η συνθήκη είναι false, οπότε η διεργασία εισβάλλει στο κρίσιμο τμήμα

Επομένως βλέπουμε ότι ο κώδικας εξασφαλίζει τον αμοιβαίο αποκλεισμό.

ΘΕΜΑ 3

ΕΡΩΤΗΜΑ Α

var s1, s2, s3: semaphores; s1=s2=s3=0;		
Διεργασία Student_1	**Διεργασία Student_2**	**Διεργασία Student_3**

Search_Book();
 	
wait(s1);
signal(s3);
signal(s3);
 	
Study_Project();
	Search_Book();
	
	wait(s2);
	signal(s1);
	wait(s3);
	
	Study_Project();
		Search_Book();
		
		signal(s2);
		wait(s3);
		
		Study_Project();

ΕΡΩΤΗΜΑ Β

```
binary semaphore mutex = 1;
binary semaphore library = 1;
binary semaphore insertion = 1;
int count = 0;
```

Searching

```
while (TRUE) {
down(mutex);
count = count + 1;
if (count == 1)
down(library);
up(mutex);
Search_Book();
down(mutex);
count = count - 1;
if (count == 0) up(library);
up(mutex);
}
```

Deletion

```
while (TRUE) {
down(library);
Delete_Book();
up(library);
}
```

Insertion

```
while(TRUE)
 {
down(mutex);
 if (count != 0)
down(insertion);
Insert_Book();
up(insertion);
up(mutex);

else ( count==0 )
down(library);
down(insertion);
Insert_Book();
up(insertion);
up(library);
up(mutex);
}
```

ΜΕΡΟΣ 4

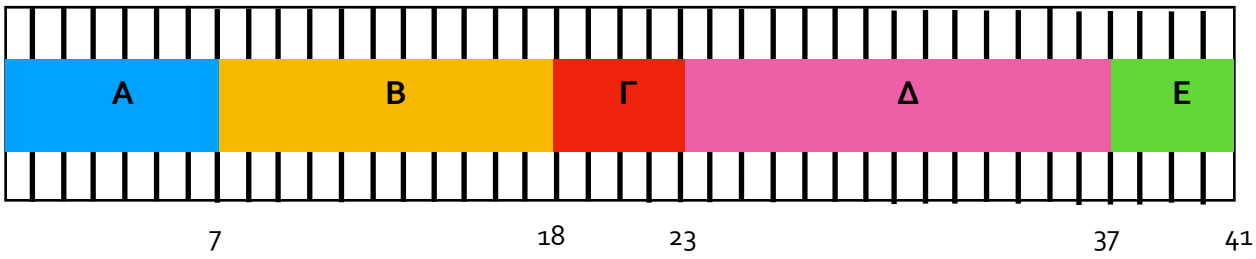
ΔΙΕΡΓΑΣΙΑ	ΧΡΟΝΙΚΗ ΣΤΙΓΜΗ ΑΦΙΞΗΣ	ΑΠΑΙΤΗΣΕΙΣ ΧΡΟΝΟΥ ΕΚΤΕΛΕΣΗΣ	ΠΡΟΤΕΡΑΙΟΤΗΤΑ
Α	0	7	2
Β	3	11	3
Γ	0	5	1
Δ	12	14	5
Ε	14	4	4

<u>Υπολογισμός Μέσων Χρόνων Διεκπεραίωσης (ΜΧΔ):</u>
Χρόνος Διεκπεραίωσης της διεργασίας $δ_i$ ($ΧΔ_i$) = (χρόνος ολοκλήρωσης της $δ_i$ – χρόνος άφιξης της $δ_i$)
Μέσος Χρόνος Διεκπεραίωσης (ΜΧΔ)= $Σ_{i=1..n}(ΧΔ_i)/n$.

<u>Υπολογισμός Μέσων Χρόνων Αναμονής (ΜΧΑ):</u>
Χρόνος Αναμονής διεργασίας $δ_i$ ($ΧΑ_i$) = (χρόνος διεκπεραίωσης της $δ_i$ – χρόνος εκτέλεσης της $δ_i$ στη ΚΜΕ)
Μέσος Χρόνος Αναμονής (ΜΧΑ)= $Σ_{i=1..n}(ΧΑ_i)/n$

Με βάση τις παραπάνω σχέσεις, οι υπολογισμοί των χρόνων διεκπεραίωσης (ΧΔ) και αναμονής (ΧΑ) των διεργασιών, για καθένα αλγόριθμο χρονοδρομολόγησης, παρουσιάζονται στον ακόλουθο πίνακα κάθε αλγορίθμου. Στην τελευταία γραμμή του κάθε πίνακα παρουσιάζονται οι μέσοι χρόνοι διεκπεραίωσης (ΜΧΔ) και οι μέσοι χρόνοι αναμονής (ΜΧΑ).

FCFS



Χρόνοι Αναμονής & Απόκρισης:

ΧΑ = ΧΟ - ΧΕκ

ΧΑΑ = 7 - 7 = 0

ΧΑΒ = 15 - 11 = 4

ΧΑΓ = 14 - 5 = 9

ΧΑΔ = 25 - 14 = 11

ΧΑΕ = 27 - 4 = 23

Χρόνος Ολοκλήρωσης:

ΧΟ = ΧΟλ - ΧΑφ

ΧΟΑ = 7 - 0 = 7

ΧΟΒ = 18 - 3 = 15

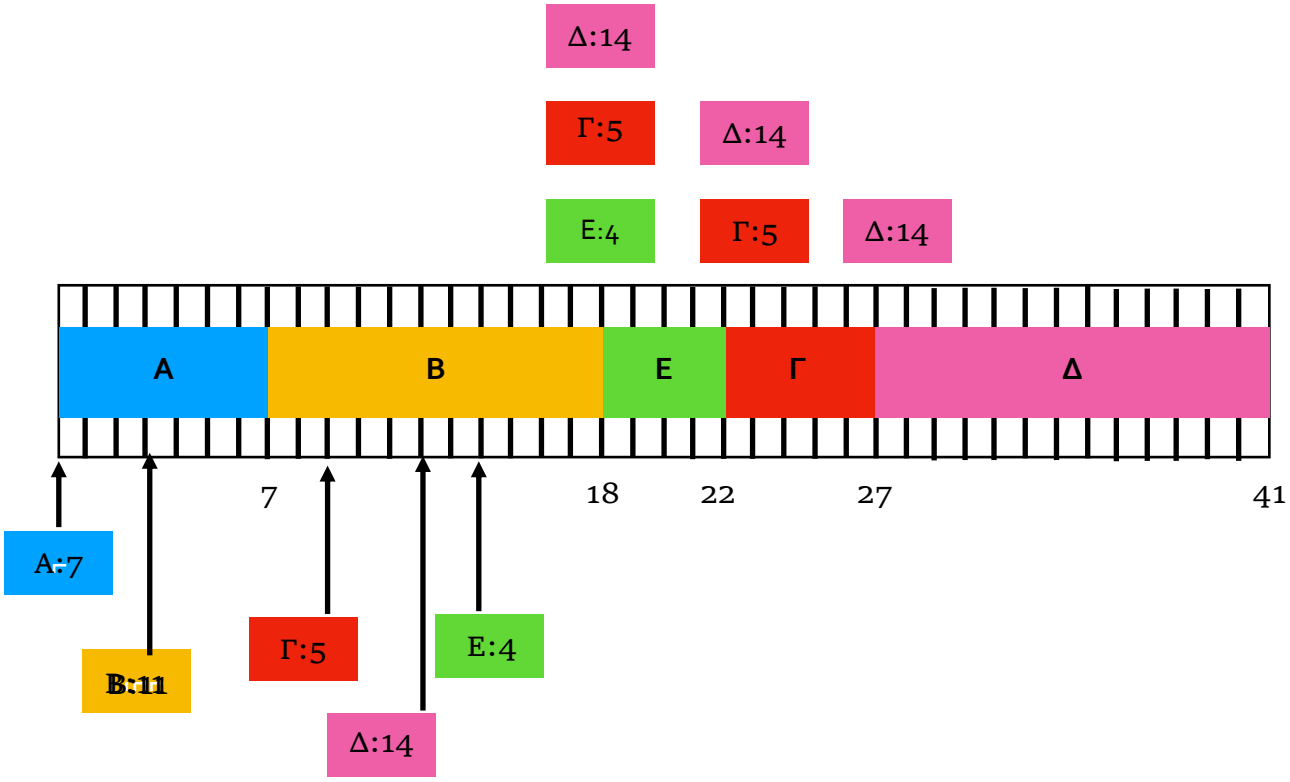
ΧΟΓ = 23 - 9 = 14

ΧΟΔ = 37 -12 = 23

ΧΟΕ = 41 -14 = 27

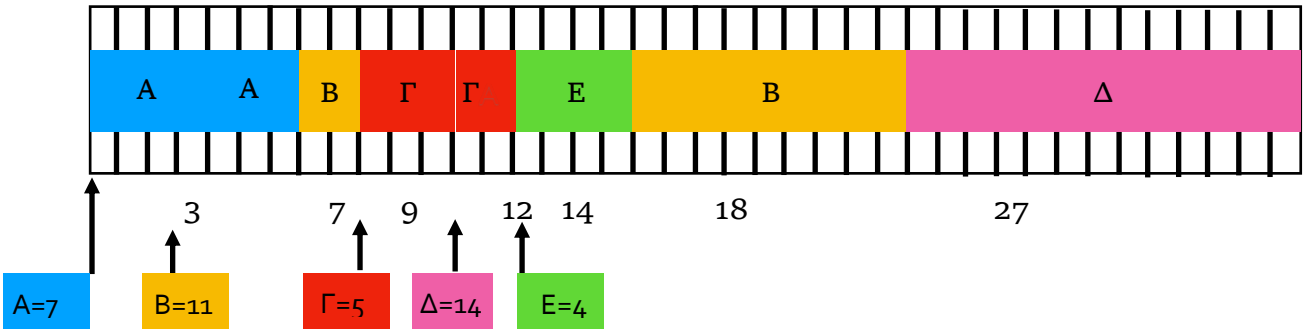
ΔΙΕΡΓΑΣΙΑ	ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ	ΧΡΟΝΟΣ ΑΠΟΚΡΙΣΗΣ	ΧΡΟΝΟΣ ΟΛΟΚΛΗΡΩΣΗΣ	ΘΕΜΑΤΙΚΕΣ ΕΝΑΛΛΑΓΕΣ
Α	0	0	7	1
Β	4	4	15	1
Γ	9	9	14	1
Δ	11	11	23	1
Ε	23	23	27	1
Μ.Ο.	9,4	9,4	17,6	1

SJF



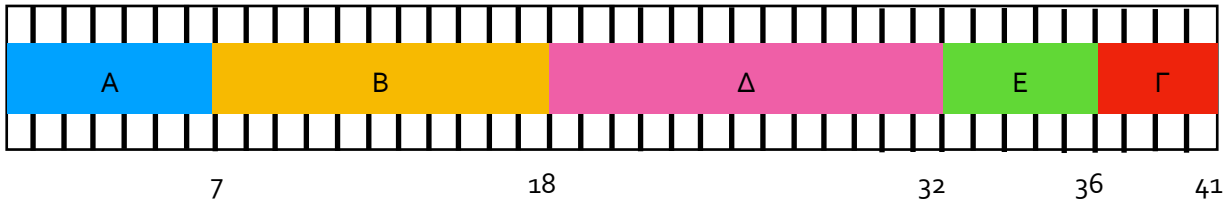
	ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ	ΧΡΟΝΟΣ ΑΠΟΚΡΙΣΗΣ	ΧΡΟΝΟΣ ΟΛΟΚΛΗΡΩΣΗΣ	ΘΕΜΑΤΙΚΕΣ ΕΝΑΛΛΑΓΕΣ
Α	0	0	7	1
Β	7-3=4	7-3=4	18-3=15	1
Γ	22-9=13	22-9=13	27-9=18	1
Δ	27-12=15	27-12=15	41-12=29	1
Ε	18-14=4	18-4=4	22-14=8	1
Μ.Ο.	7,2	7,2	15,4	1

SRTF



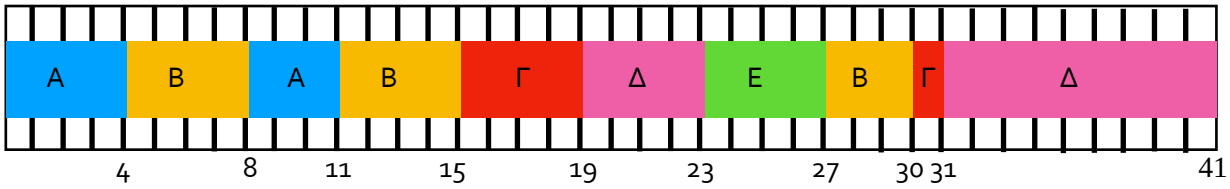
ΔΙΕΡΓΑΣΙΑ	ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ	ΧΡΟΝΟΣ ΑΠΟΚΡΙΣΗΣ	ΧΡΟΝΟΣ ΟΛΟΚΛΗΡΩΣΗΣ	ΘΕΜΑΤΙΚΕΣ ΕΝΑΛΛΑΓΕΣ
Α	0	0	7	1
Β	(7-3)+(18-9)= 13	7-3=4	27-3=24	2
Γ	0	0	14-9=5	1
Δ	0	0	18-14=4	1
Ε	27-12=15	15	41-12=29	1
Μ.Ο.	5,6	3,8	13,8	1,2

PRIORITY SCHEDULING



ΔΙΕΡΓΑΣΙΑ	ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ	ΧΡΟΝΟΣ ΑΠΟΚΡΙΣΗΣ	ΧΡΟΝΟΣ ΟΛΟΚΛΗΡΩΣΗΣ	ΘΕΜΑΤΙΚΕΣ ΕΝΑΛΛΑΓΕΣ
Α	0	0	7	1
Β	7-3=4	7-3=4	18-3=15	1
Γ	36-9=27	36-9=27	41-9=32	1
Δ	18-12=6	18-12=6	32-12=20	1
Ε	32-14=18	32-14=18	41-12=29	1
Μ.Ο.	11	11	19,2	3

RR



ΔΙΕΡΓΑΣΙΑ	ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ	ΧΡΟΝΟΣ ΑΠΟΚΡΙΣΗΣ	ΧΡΟΝΟΣ ΟΛΟΚΛΗΡΩΣΗΣ	ΘΕΜΑΤΙΚΕΣ ΕΝΑΛΛΑΓΕΣ
Α	4	0	11-0=11	2
Β	(4-3)+(11-8)+(27-15)=16	4-3=1	30-3=27	3
Γ	(15-9)+(30-19)=17	15-9=6	31-9=22	2
Δ	(19-12)+(31-23)=15	19-12=7	41-12=29	2
Ε	23-14=9	23-14=9	27-14=13	1
Μ.Ο.	12,2	4,6	19,2	3