ΕΥΣΤΑΘΙΟΣ ΙΩΣΗΦΙΔΗΣ

Άσκηση 1

Δίνεται ένα σύστημα με 3 παραγωγούς και ένα καταναλωτή.

Κάθε διεργασία παραγωγός έχει ένα κρίσιμο τμήμα που τοποθετεί ένα στοιχείο σε μία μνήμη. Ο καταναλωτής έχει ένα κρίσιμο τμήμα που καταναλώνει τα *τρία* στοιχεία που τοποθετούν οι παραγωγοί. Ο καταναλωτής μπορεί να μπει σε κρίσιμο τμήμα μόνο όταν εκτελεστούν και οι τρεις παραγωγοί. Οι παραγωγοί (εκτός της πρώτης εκτέλεσης) μπαίνουν σε κρίσιμο τμήμα με τη σειρά Παραγωγός 1, Παραγωγός 2, Παραγωγός 3 αφότου καταναλωθούν τρία στοιχεία από τον καταναλωτή.

Producer 1	Producer 2	Producer 3	Consumer
:	:	:	:
AddItem()	AddItem()	AddItem()	ConsumeItem()
:	:	:	:

Να ορίσετε σηματοφορείς για το παραπάνω σύστημα.

ΛΥΣΗ

Αρχικές τιμές σηματοφορέων: mutex = 1, P1= 1, P2 = P3 = C =0

Producer 1

```
while(true){
   down(P1)
   down(mutex)
   AddItem()
   up(mutex)
   up(P2)
}
```

Producer 2

```
while(true){
   down(P2)
   down(mutex)
   AddItem()
   up(mutex)
   up(P3)
}
```

Producer 3

```
while(true){
   down(P3)
   down(mutex)
   AddItem()
   up(mutex)
   up(C)
}
```

Consumer

```
while(true){
   down(C)
   down(mutex)
   ConsumeItem()
   up(mutex)
   up(P1)
}
```

ΕΥΣΤΑΘΙΟΣ ΙΩΣΗΦΙΔΗΣ

Άσκηση 2

Έστω ότι δύο διεργασίες Δ1 και Δ0 τρέχουν με σταθερά κβάντα 100 χρονικές μονάδες και χρησιμοποιούν τη λύση του Peterson, όπως φαίνεται παρακάτω. Έστω ότι ξεκινάει αρχικά η Δ0 και οι 100 μονάδες αρκούν μέχρι να τεθεί interested[process]=true (πριν αλλάξει η turn). Για τη Δ1, οι 100 μονάδες αρκούν για να μπει στο βρόχο while της εισαγωγής στο κρίσιμο τμήμα. Σε ποια χρονική στιγμή θα μπουν σε ΚΤ οι δύο διεργασίες (αν θα μπουν); Ξεκινήστε από το χρόνο t=0.

```
#define FALSE  0
#define TRUE   1
#define N      2                        /* number of processes */

int turn;                               /* whose turn is it? */
int interested[N];                      /* all values initially 0 (FALSE) */

void enter_region(int process);         /* process is 0 or 1 */
{
     int other;                         /* number of the other process */

     other = 1 – process;               /* the opposite of process */
     interested[process] = TRUE;        /* show that you are interested */
     turn = process;                    /* set flag */
     while (turn == process && interested[other] == TRUE) /* null statement */ ;
}

void leave_region(int process)          /* process: who is leaving */
{
     interested[process] = FALSE;       /* indicate departure from critical region */
}
```

ΛΥΣΗ

t: 0 – 100 (Δ0)	process = Δ0 other = Δ1 (το process είναι το Δ0, το other δείχνει την άλλη διεργασία) interested[Δ0] = true (εδώ κόβεται η Δ0)
t: 100 – 200 (Δ1)	process = Δ1 other = Δ0 interested[Δ1] = true turn = Δ1 (αλλάζει η σειρά και μπαίνει η επόμενη διεργασία. Μπαίνει στο while αλλά δεν προλαβαίνει να ελέγξει)
t: 200 – 300 (Δ0)	turn = Δ0 (πριν ήταν Δ1 άρα αλλάζει) μπαίνει στο while (Δ0==Δ0 && interested[Δ1]==true) το οποίο βγαίνει *true* άρα, ΔΕΝ μπαίνει στο κρίσιμο τμήμα
t: 300 – 400 (Δ1)	Ελέγχει το while (Δ0 == Δ1 && interested[Δ0]==true) το οποίο βγαίνει false, άρα μπαίνει στο **κρίσιμο τμήμα** και αλλάζει το interested[Δ1] = false στο leave_region
t: 400 – 500 (Δ0)	Λόγω της προηγούμενης αλλαγής, το while (Δ0 == Δ0 && interested[Δ1]==false) δίνει false, άρα η Δ0 μπαίνει στο **κρίσιμο τμήμα**.

ΕΥΣΤΑΘΙΟΣ ΙΩΣΗΦΙΔΗΣ

Άσκηση 3

Να γράψετε μία επέκταση του προβλήματος παραγωγού καταναλωτή, στο οποίο υπάρχουν 2 καταναλωτές και ισχύουν οι εξής κανόνες:

1) Ο παραγωγός πρέπει να γράφει στη μνήμη αφότου γίνει ανάγνωση από τους καταναλωτές, με εξαίρεση την πρώτη εγγραφή

2) Κάθε στοιχείο που παράγεται πρέπει να καταναλώνεται και από τους δύο καταναλωτές με τη σειρά Καταναλωτής 1, Καταναλωτής 2

3) Οι καταναλωτές δεν μπορούν να διαβάζουν ταυτόχρονα

Process Producer	Process Consumer 1	Process Consumer 2
:	:	:
repeat	repeat	repeat
add_item	consume_item	consume_item
:	:	:
until forever	until forever	until forever

ΛΥΣΗ

Κάθε διεργασία έχει δικό της σηματοφορέα. Θα είναι αντίστοιχα:
P: για τον producer
C1: για τον καταναλωτή Consumer 1
C2: για τον καταναλωτή Consumer 2
Mutex: χρησιμοποιείται για να κλειδώνει την μνήμη
Θεωρούμε ότι ο παραγωγός, παράγει 2 στοιχεία. Δεν μετράμε τις γεμάτες ή κενές θέσεις (δεδομένου ότι ο παραγωγός παράγει 2 στοιχεία και ο κάθε καταναλωτής καταναλώνει από 1).
Αρχικές τιμές σημαφορέων mutex = 1, P = 1, C1 = C2 = 0

Process Producer	**Process Consumer 1**	**Process Consumer 2**
while(true){	while(true){	while(true){
down(P)	down(C1)	down(C2)
down(mutex)	down(mutex)	down(mutex)
add_item	consume_item	consume_item
up(mutex)	up(mutex)	up(mutex)
up(C1)	up(C2)	up(P)
}	}	}

ΕΥΣΤΑΘΙΟΣ ΙΩΣΗΦΙΔΗΣ

Άσκηση 4

Με δεδομένη την (γενικά εσφαλμένη) παραδοχή, ότι καθεμία από τις διεργασίες θα λαμβάνει τα κατάλληλα κβάντα, έτσι ώστε να εκτελεστεί ως το τέλος της (δηλαδή ΔΕΝ θα κοπεί στη μέση), θεωρήστε ένα σύστημα, το οποίο εκτελεί 5 διεργασίες. Να ορίσετε τους σηματοφορείς και τον τρόπο χρησιμοποίησης των λειτουργιών up/down (χωρίς να αλλάξετε με άλλον τρόπο τον κώδικα), έτσι ώστε να είναι δυνατή η επαναληπτική εκτύπωση της συμβολοσειράς

ADBBBBCCCEE

Διεργασία 1	Διεργασία 2	Διεργασία 3	Διεργασία 4	Διεργασία 5
....	....	....	....	....
Επανάλαβε	Επανάλαβε	Επανάλαβε	Επανάλαβε	Επανάλαβε
....	....	....	....	....
Τύπωσε Α	Τύπωσε Β	Τύπωσε Γ	Τύπωσε Δ	Τύπωσε Ε
.....	.....	.....	.....	.....
Έως (για πάντα)	Έως (για πάντα)	Έως (για πάντα)	Έως (για πάντα)	Έως (για πάντα)

ΛΥΣΗ

Θέλουμε να εκτυπώσει με την σειρά **ADBBBBCCCEE**

Υποθέτουμε τους σηματοφορείς A = 2, B = C = D = E = 0

Διεργασία Α	**Διεργασία Β**	**Διεργασία C**	**Διεργασία D**	**Διεργασία Ε**
down(A)	down(B)	down(C)	down(D)	down(E)
down(A)	Τύπωσε Β	down(C)	Τύπωσε Δ	down(E)
Τύπωσε Α	up(C)	down(C)	up(B)	down(E)
up(D)	up(C)	down(C)	up(B)	Τύπωσε Ε
	up(C)	Τύπωσε Γ	up(B)	up(A)
		up(E)	up(B)	
		up(E)		

Έστω ότι A = 2 και όλες οι άλλες κοιμούνται (B = C = D = E = 0).

A: Ξεκινάει η Α και τυπώνει 1 φορά. Μηδενίζει το Α με τις 2 down και τυπωνει. Αυξάνει κατά ένα το D και την ξυπνάει. Άρα οι σηματοφορείς θα είναι D = 1, A = B = C = E = 0. **ΕΚΤΥΠΩΣΗ: Α**

D: Ξεκινάει η D και τυπώνει μια φορά. Μηδενίζει το D (με το down). Με τα 4 up, αυξάνει το B = 4 και το ξυπνάει. Άρα οι σηματοφορείς θα είναι B = 4, A = C = D = E = 0. **ΕΚΤΥΠΩΣΗ: D**

B: Ξεκινάει η Β και τυπώνει 4 φορές. Με τις 4 down η Β μηδενίζει ενώ η C αυξάνεται στο 12 και ξυπνάει. Άρα οι σηματοφορείς θα είναι C = 12, A = B = D = E = 0. **ΕΚΤΥΠΩΣΗ: BBBB**

C: Ξεκινάει η C και τυπώνει 3 φορές. Επειδή έγιναν 4 επαναλήψεις με 3 up(C), το C=12 για να τυπωθεί 3 φορές, χρειάζονται 3 επαναλήψεις, άρα και 4 down. Οι επαναλήψεις εκτελούν τα εξής:

12 έως 8, εκτύπωση C 2 * up(E) => E=2

8 έως 4, εκτύπωση C 2 * up(E) => E=2+2 = 4

4 έως 0, εκτύπωση C 2 * up(E) => E=4+2 = 6

ΚΟΙΜΑΤΑΙ

ΑΡΑ

E=6

Άρα οι σηματοφορείς θα είναι E = 6, A = B = C = D = 0. **ΕΚΤΥΠΩΣΗ: CCC**

E: Ξεκινάει η Ε για να τυπώσει 2 φορές. Επειδή E=6 χρειάζομαι 2 επαναλήψεις με 3 down(E). Στο τέλος οι σηματοφορείς θα έχουν τις αρχικές τιμές τους, δηλαδή A = 2, B = C = D = E = 0 **ΕΚΤΥΠΩΣΗ: ΕΕ**

ΕΥΣΤΑΘΙΟΣ ΙΩΣΗΦΙΔΗΣ

Άσκηση 5

Δίνονται *2 διεργασίες εκτύπωσης*, οι οποίες εκτελούνται ανά *200 ms* και χρησιμοποιούν TSL. Ο κώδικας του τμήματος εκτύπωσης (**κρίσιμο τμήμα**) είναι ο εξής:

Διάβασε τη μεταβλητή In
Γράψε στο Spool Directory
Θέσε In=In+1

Έστω ότι τα κβάντα και των 2 διεργασιών φτάνουν για να διαβάσουν τη μεταβλητή In, η οποία πριν την εκκίνηση της εκτέλεσης των διεργασιών έχει τιμή 5. Να δώσετε τις τιμές των Flag, Reg και In, όταν μία από τις διεργασίες καταφέρει να μπει σε κρίσιμο τμήμα.

ΛΥΣΗ

Έστω έχω 2 διεργασίες την Α και την Β.

```
1 enter_region:
2 TSL REGISTER,LOCK
3 CMP REGISTER,#0
4 JNE enter_region
5 RET

6 leave_region:
7 MOVE LOCK,#0
8 RET
```

Σε χρόνο t: 0-200
Μπαίνει η Α. Έστω ότι LOCK = 0.
Στην γραμμή 2 θα έχω REG = 0 και LOCK = 1.
Στην γραμμή 3 θα συγκρίνει το REG με το 0 (αληθές) και
έτσι μπαίνει στο **κρίσιμο τμήμα,** διαβάζει την **In = 5** και κόβεται.

Σε χρόνο t: 200-400
Μπαίνει η Β. Βλέπει την LOCK = 1
Στην γραμμή 2 θα έχω REG = 1 και LOCK = 1.
Άρα επαναλαμβάνει την TSL και ΔΕΝ μπαίνει στο κρίσιμο τμήμα μέχρι να τελειώνουν τα κβάντα.

Σε χρόνο t: 400-600
Έρχεται η σειρά της Α. Έχει διαβάσει ήδη **In = 5** και γράφει στο Spool Directory και κάνει την **In = 6**
Βγαίνει από το κρίσιμο τμήμα και αλλάζει την LOCK = 0 στην γραμμή 7 του leave_region.

ΕΥΣΤΑΘΙΟΣ ΙΩΣΗΦΙΔΗΣ

Άσκηση 6

Να γράψετε μία επέκταση του προβλήματος παραγωγού- καταναλωτή, στο οποίο υπάρχουν 2 παραγωγοί και 3 καταναλωτές και η σειρά εκτέλεσης είναι: Π1, Π2 και τυχαία σειρά κατανάλωσης. Μόλις καταναλώσουν όλοι οι καταναλωτές, αρχίζει ξανά ο Π1.

ΛΥΣΗ

Κάθε διεργασία έχει δικό της σηματοφορέα. Θα είναι αντίστοιχα:
P1: για τον Π1
P2: για τον Π2
C1: για τον καταναλωτή 1
C2: για τον καταναλωτή 2
C3: για τον καταναλωτή 3
Mutex: χρησιμοποιείται για να κλειδώνει την μνήμη
Αρχικές τιμές σηματοφορέων: mutex = 1, P1= 3, P2 = C1 = C2 = C3 =0

Παραγωγός 1

```
while(true){
   down(P1)
   down(P1)
   down(P1)
   down(mutex)
   AddItem()
   up(mutex)
   up(P2)
}
```

Παραγωγός 2

```
while(true){
   down(P2)
   down(mutex)
   AddItem()
   up(mutex)
   up(C1)
   up(C2)
   up(C3)
}
```

Καταναλωτής 1

```
while(true){
   down(C1)
   down(mutex)
   ConsumeItem()
   up(mutex)
   up(P1)
}
```

Καταναλωτής 2

```
while(true){
   down(C2)
   down(mutex)
   ConsumeItem()
   up(mutex)
   up(P1)
}
```

Καταναλωτής 3

```
while(true){
   down(C3)
   down(mutex)
ConsumeItem()
   up(mutex)
   up(P1)
}
```

ΕΥΣΤΑΘΙΟΣ ΙΩΣΗΦΙΔΗΣ

Άσκηση 7

Να εξετάσετε πιθανά σενάρια διακοπής των κβάντων όταν χρησιμοποιείται η λύση Peterson και TSL. Να ελέγξετε τις περιπτώσεις ενεργού αναμονής και αμοιβαίου αποκλεισμού.