ΥΠΟΥΡΓΕΙΟ ΕΘΝΙΚΗΣ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ ΕΠΕΑΕΚ
ΕΥΡΩΠΑΪΚΗ ΕΝΩΣΗ
ΣΥΓΧΡΗΜΑΤΟΔΟΤΗΣΗ
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Η ΠΑΙΔΕΙΑ ΣΤΗΝ ΚΟΡΥΦΗ
Επιχειρησιακό Πρόγραμμα
Εκπαίδευσης και Αρχικής
Επαγγελματικής Κατάρτισης
ΕΛΛΑΔΑ 2008
Ανάπτυξη παντού. Ανάπτυξη για όλους.

Κατανεμημένος Υπολογισμός
Παναγιώτα Φατούρου

2ο Σετ Ασκήσεων

Προθεσμία Παράδοσης: Τετάρτη 20/6/07 , στη θυρίδα μου

Άσκηση 1 (20 μονάδες)

α. Παρουσιάστε σενάριο εκτέλεσης του ανιχνευτή συμφόρησης (contention detector, σελ. 24 σετ διαφανειών Mutual Exclusion) στο οποίο καμία διεργασία δεν επιστρέφει win.

β. Αποδείξτε ότι ο μη-συμμετρικός αλγόριθμος αμοιβαίου αποκλεισμού για 2 διεργασίες (σελ. 14 σετ διαφανειών Mutual Exclusion) επιτυγχάνει αμοιβαίο αποκλεισμό και αποφυγή αδιεξόδου.

γ. Εξηγήστε πως λειτουργεί ο Tournament αλγόριθμος όταν ο αριθμός των διεργασιών δεν είναι δύναμη του 2.

δ. Παρουσιάστε φορμαλιστική απόδειξη για το γενικευμένο λήμμα (Λήμμα 2) της διαφάνειας 32 του σετ διαφανειών Mutual Exclusion.

Άσκηση 2 (80 μονάδες)

Σας δίνεται η εξής παραλλαγή του αλγορίθμου του αρτοπωλείου (Bakery Algorithm):

Shared Variables: **// Κώδικας για διεργασία p_i**

```
direction: a bit of type {LEFT, RIGHT};
choosing[1..n]: bolean array;
mydirection[1..n]: array of type {LEFT,RIGHT};
number[1..n]: array of type {0,..,n}

// initially, for each j, 1 ≤ j ≤ n, choosing[j] = FALSE, number[j] = 0;
// the initial values of all other variables are immaterial
```

<Entry>:

```
1: choosing[i]=TRUE;                    //interested in entering critical section
2: mydirection[i] = direction;
3: number[i] = max({number[j] | (1 ≤ j ≤ n) && mydirection [j] == mydirection[i]}) + 1;
4: choosing[i]=FALSE;
5: for j=1 to n do  {
6:       wait until choosing[j] == FALSE;
7:       if (mydirection[i] == mydirection[j])
8:           wait until (number[j]==0 || (number[j],j)>(number[i],i) || mydirection[j]≠mydirection[i]);
```

```
9:      else
10:         wait until (number[j]==0 || mydirection[i]≠direction || mydirection[j]==mydirection[i]);
}

<Critical Section>;

<Exit>:
11: if (mydirection[i] == LEFT) direction = RIGHT;
12: else direction = LEFT;
13: number[i]=0;

<Remainder>;
```

α. Εξηγήστε πως λειτουργεί ο παραπάνω αλγόριθμος (δηλαδή παρουσιάστε διαισθητική περιγραφή του αλγορίθμου).

β. Αποδείξτε πως ο παραπάνω αλγόριθμος είναι μια σωστή λύση στο πρόβλημα του αμοιβαίου αποκλεισμού.

Υπόδειξη: Ίσως σας βοηθήσει να ξεκινήσετε αποδεικνύοντας τα παρακάτω λήμματα:

Λήμμα 1: Έστω ότι σε κάποια χρονική στιγμή t, η τιμή της μεταβλητής direction είναι d ∈ {LEFT, RIGHT}. Τότε, κάθε διεργασία p_j, που τη χρονική στιγμή t βρίσκεται στο τμήμα εισόδου (entry) και για την οποία ισχύει ότι mydirection[j] ≠ d, πρέπει να εισέλθει στο κρίσιμο τμήμα πριν εισέλθει οποιαδήποτε διεργασία με mydirection == d.

Λήμμα 2: Έστω ότι σε κάποια χρονική στιγμή t η τιμή της μεταβλητής direction αλλάζει από d ∈ {LEFT, RIGHT} στην άλλη τιμή. Τότε, για οποιαδήποτε διεργασία p_j βρίσκεται στο τμήμα εισόδου στην t ισχύει ότι mydirection[j] == d.

γ. Οδηγεί η παραπάνω λύση σε παρατεταμένη στέρηση (starvation);

δ. Ποια είναι η χωρική πολυπλοκότητα του παραπάνω αλγορίθμου (δηλαδή πόσοι καταχωρητές χρησιμοποιούνται και πόσο μεγάλο είναι το σύνολο τιμών που αποθηκεύει ο καθένας από αυτούς);

ε. Τι θα συμβεί αν η γραμμή 13 του αλγορίθμου τοποθετηθεί πριν από την γραμμή 11. Θα είναι το αποτέλεσμα σωστή λύση στο πρόβλημα του αμοιβαίου αποκλεισμού; Γιατί ναι, γιατί όχι;

στ. Τι θα συμβεί αν παραληφθεί η τρίτη συνθήκη της γραμμής 10; Θα είναι το αποτέλεσμα σωστή λύση στο πρόβλημα του αμοιβαίου αποκλεισμού; Γιατί ναι, γιατί όχι;