**BÀI KIỂM TRA ÔN TẬP CHƯƠNG 5**

*(thời gian làm bài 45 phút)*

**Câu 1**: Xét giải pháp đồng bộ hóa sau có bảo đảm 3 điều kiện của một bài toán giải quyết tranh chấp hay không? Giải thích bằng 1 kịch bản thực thi giữa 2 tiến trình thực thi đồng thời. *(Gợi ý: giả sử 2 tiến trình Pi và Pj với i = 0 và j =1 có cùng đoạn code như bên dưới, thay i và j tương ứng với tiến trình Pi và Pj. Sau đó giả sử 1 trường hợp thực thi mà 1 tiến trình đang trong vùng tranh chấp, xem tiến trình còn lại có vào được hay không; và giả sử 1 tiến trình đang ở ngoài vùng tranh chấp có ngăn cản tiến trình còn lại vào vùng tranh chấp hay không?)*

|  |
| --- |
| *boolean flag[2];*  *int i, j;*  *while (TRUE)*  *{*  *turn = j;*  *flag[i] = true;   while (flag[j] and turn == j);*  ***critical section***  *flag[i] = false;*  ***remainder section***  *};* |

**Trả lời:**

- Giải thuật này không thoả mãn:

* Loại trừ tương hỗ: tiến trình Pj chỉ vào vùng tranh chấp khi flag[i]=false.

Giả sử Pi đang trong vùng tranh chấp, tức là flag[i]=true (được gán). Do đó P[j] vẫn có thể vào vùng tranh chấp

* Progress: turn thay đổi trước khi vào vùng tranh chấp. Khi Pj muốn vào vùng tranh chấp thì flag[i]=true và Pi vào cùng tranh chấp ngay lập túc.

Khi cả 2 muốn vào vùng tranh chấp và turn = j. Cả flag[i] và flag[j] đều bằng true, nhưng khi Pj tạm dừng thì lại thiết lập biến flag[i] = false và ngăn cản Pi vào vùng tranh chấp.

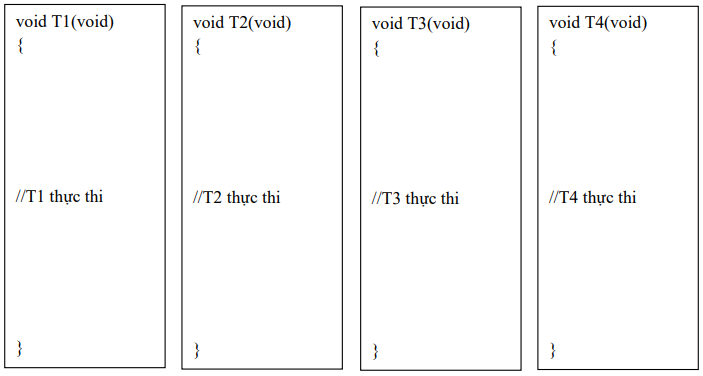
* Chờ đợi giới hạn: Pi có thể sẽ bị chờ vô hạn định vì biến turn không được thiết lập lại bằng I sau mỗi lần Pj vào và ra khỏi vùng tranh chấp.

Xét tình huống khi flag[0] =1; turn =0; lúc này P0 vào CS. Nếu lúc đó flag[1] = 1, P1 có thể gán turn = 1 và vào luôn CS (2 tiến trình cùng vào CS một lúc).

**Câu 2**: Xét một hệ thống có 4 tiểu trình T1, T2, T3, T4. Quan hệ giữa các tiểu trình này được biểu diễn như sơ đồ bên dưới, với mũi tên từ tiểu trình (Tx) sang tiểu trình (Ty) có nghĩa là tiểu trình Tx phải kết thúc quá trình hoạt động của nó trước khi tiểu trình Ty bắt đầu thực thi. Giả sử tất cả các tiểu trình đã được khởi tạo và sẵn sàng để thực thi. Hãy sử dụng semaphore để đồng bộ hoạt động của các tiểu trình sao cho đúng với sơ đồ đã cho bên dưới.



1. Khai báo và khởi tạo các semaphore
2. Đặt các lệnh wait() và signal() của từng semaphore vào hàm thực thi của từng tiểu trình để đảm bảo yêu cầu đồng bộ trên như trong hình bên dưới



**Trả lời:**

1. Khai báo và khởi tạo các semaphone

Init(sem1,0); // Khởi tạo semaphore sem1 có giá trị bằng 0

Init(sem2,0); // Khởi tạo semaphore sem2 có giá trị bằng 0

1. Thực thi

void T1(void)

{

// T1 thực thi

Signal(sem1)

}

void T2(void)

{

wait(sem1)

// T2 thực thi

signal(sem2)

signal(sem2)

}

void T3(void)

{

wait(sem2)

// T3 thực thi

}

void T4(void)

{

wait(sem2)

// T4 thực thi

}