**BÀI TẬP THỰC HÀNH SỐ 4**

# Câu hỏi 1: Chạy chương trình trên vài lần. Bạn nhận thấy điều gì? Giải thích!

Chạy lần 1:



Chạy lần 2:



Mỗi lần chạy sẽ có thể ra một kết quả nhỏ hơn 3000. Bởi vì ở đây ta chạy bất đồng bộ nên việc truy cập vào biến rsc là tùy thuộc vào worker nào nhanh hơn, nên việc biến rsc thay đổi phụ thuộc vào việc truy cấp biến rsc của các worker.

# Câu hỏi 2:

Chạy lần nào kết quả cũng là 3000. Ở đây ta đã thêm cơ chế đồng bộ.

# Câu hỏi 3:

Không có gì khác nhau cả. Biến rsc là tài nguyên găng, khi có 1 thread sử dụng, các thread khác không thể sử dụng biến rsc và phải đợi biến rsc sử dụng xong.

# Câu hỏi 4:

Code:

void \*fun(void \*args)

{

    time\_t start = time(NULL);

    time\_t end = start + 5; //run for 5 seconds

    ++shared;

    return NULL;

}

Kết quả chạy:



# Câu hỏi 5:

Kết quả khác nhau:

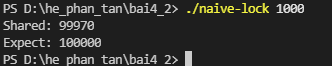


Giải thích:

Vì có nhiều luồng vào đoạn găng tại 1 thời điểm

# Câu hỏi 6:

Kết quả khác nhau:



Giải thích: việc nắm giữ lock lâu sẽ dẫn đến bị ngắt

# Câu hỏi 7:

Coarse Locking sẽ khóa chương trình bằng cách sử dụng duy nhất 1 lock cho toàn bộ đoạn găng. Cho nên việc truy cập vào tài nguyên găng sẽ được kiểm soát.

# Câu hỏi 8: so sánh và đo đạt thời gian để chứng minh là Fine Locking sẽ nhanh hơn Coarse Locking.

Thời gian chạy của Fine locking chậm hơn Coarse locking

# Câu hỏi 9: chạy chương trình trên và bạn nhận thấy điều gì? Giải thích thông qua việc phân tích mã nguồn.

Chương trình bị treo. Bởi vì trong đoạn code có 2 đoạn

pthread\_mutex\_lock(&lock\_a); //lock a then b

        pthread\_mutex\_lock(&lock\_b);

        //CRITICAL SECTION

        a++;

        b++;

        pthread\_mutex\_unlock(&lock\_a);

        pthread\_mutex\_unlock(&lock\_b);

  pthread\_mutex\_lock(&lock\_b); //lock b then a

        pthread\_mutex\_lock(&lock\_a);

        //CRITICAL SECTION

        a++;

        b++;

        pthread\_mutex\_unlock(&lock\_b);

        pthread\_mutex\_unlock(&lock\_a);

ở đây cả 2 đoạn đều truy cập vào tài nguyên găng, một đoạn truy cập vào biến a, một đoạn truy cập vào biến b trước =>việc đợi chờ tài nguyên để sử dụng => treo chương trình.