Tên: Phan Ngọc Hạnh Nhi

MSSV: 2131209002

**Lab 2**

1. Assignment 1a:

* Em tạo 2 threads (luồng) và cho cả 2 threads cùng truy cập vào dữ liệu chung, trong đó:
* Thread 1 sẽ tăng giá trị của dữ liệu chung lên 1.000.000 lần.
* Thread 2 sẽ giảm giá trị của dữ liệu chung đi 1.000.000 lần.
* Để đảm bảo thứ tự thực hiện, em cho Thread 1 chạy trước và dùng lệnh join để chờ Thread 1 hoàn thành. Sau khi Thread 1 hoàn thành xong, em mới cho Thread 2 bắt đầu chạy và cũng dùng lệnh join để chờ Thread 2 hoàn thành trước khi tiếp tục.
* Khi Thread 1 hoàn thành, giá trị của dữ liệu chung sẽ là 1.000.000 (vì tăng từ 0 lên 1.000.000).
* Sau khi Thread 2 chạy, giá trị của dữ liệu chung sẽ giảm từ 1.000.000 xuống lại 0.
* Kết quả cuối cùng của dữ liệu chung là 0 vì 2 threads chạy theo thứ tự tuần tự, không có sự xung đột luồng (race condition).

1. Assignment 1b:

* Em cũng tạo 2 threads và cho cả 2 threads cùng truy cập vào dữ liệu chung, trong đó:
* Thread 1 sẽ tăng giá trị của dữ liệu chung lên 1.000.000 lần.
* Thread 2 sẽ giảm giá trị của dữ liệu chung đi 1.000.000 lần.
* Tuy nhiên, lần này em cho cả 2 threads chạy đồng thời (song song) mà không áp dụng cơ chế đồng bộ hóa (synchronization). Do đó, khi một thread đang thực hiện công việc, thread còn lại có thể "nhảy vào" CPU để thực hiện các thao tác của nó. Điều này gây ra hiện tượng race condition (xung đột luồng), trong đó thứ tự thực hiện các thao tác tăng/giảm không được đảm bảo.
* Kết quả cuối cùng của dữ liệu chung trong trường hợp này sẽ không xác định. Giá trị có thể là:
* Một số dương, nếu các thao tác tăng nhiều hơn các thao tác giảm.
* Một số âm, nếu các thao tác giảm nhiều hơn các thao tác tăng.
* Gần 0, nhưng hiếm khi chính xác là 0.
* Lý do là các thao tác tăng (++) và giảm (--) không phải là toán tử nguyên tử (atomic operations), tức là chúng bao gồm nhiều bước (đọc giá trị, thay đổi giá trị, ghi lại giá trị). Khi 2 threads chạy đồng thời, thao tác của thread này có thể bị ghi đè bởi thread kia, dẫn đến kết quả không đúng như mong đợi.
* Kết luận:
* Assignment 1a: Kết quả luôn chính xác là 0 vì 2 threads được chạy tuần tự, không có xung đột luồng.
* Assignment 1b: Kết quả không xác định vì 2 threads chạy song song, dẫn đến hiện tượng race condition khi cả 2 cùng truy cập vào dữ liệu chung mà không có sự đồng bộ hóa.