HW4 Report

Q1. HW4_1_1

將 mutex 拿掉之後，會發現結果很明顯是錯的，因為在 data + 1 的時候其他 thread 也有可能同時在做 +1 ，導致 data race 的問題，造成輸出與正確答案來的小。

Q2. HW4_1_2

這題與上一題一樣，mutex 拿掉後會發生 data race 問題，導致結果不正確，比正確答案還要小。

Q3. HW4_2 Monte Carlo – Pi

在這題如果不使用 mutex 去確保 answer 的正確性，會發生 data race 的問題，使在圓內的點沒有正確更新到 ans 變數當中，比應該要在裡面的少。

Q4. Learning & Reflections

這次作業如果熟悉 multi thread 的程式的話，能夠非常快就完成這次的作業。但是我覺得可以讓我們試試寫看看 multi thread 搭配 local answer ，來達到 multi thread speed up 的感覺，會比這次用 lock 確保 data race 的程式來的有趣。但經過這次作業，也更熟悉 lpthread 的用法，在之後就可以自己嘗試寫看看更多平行程式。

```
./hw4_1_1.o < ./testcase/12000-1.txt
0: 3316
1: 3285
2: 3292
./hw4_1_2.o < ./testcase/12000-2.txt
0: 3199
1: 3188
2: 3209
./pi.o < ./testcase/2.input
Get: 31929657
Pi: 1.27719
```

```
./hw4_1_1.o < ./testcase/12000-1.txt
0: 4044
1: 3973
2: 3983
./hw4_1_2.o < ./testcase/12000-2.txt
0: 4044
1: 3973
2: 3983
./pi.o < ./testcase/2.input
Get: 78534471
Pi: 3.14138
```

左圖為沒有 mutex 確保沒有 data race 的實驗結果，而右邊為有用 mutex 之版本，可以看到左邊答案都比正確答案小，而右邊有使用 mutex 就有正確的數值。