OS HW4 report

Name: 李柏毅
Student ID: 0717001

Q1:
Compare results between hw4_1_1 with/without synchronization.

```
[c0717001@linux1 hw_4]$ g++ hw4_1_1.cpp -o hw4_1_1 -lpthread
[c0717001@linux1 hw_4]$ ./hw4_1_1 < ./test_case/12000-1.txt > ./test_case/myans_1.txt
[c0717001@linux1 hw_4]$ diff ./test_case/ans.txt ./test_case/myans_1.txt
[c0717001@linux1 hw_4]$
[c0717001@linux1 hw_4]$ g++ hw4_1_1.cpp -o hw4_1_1 -lpthread
[c0717001@linux1 hw_4]$ ./hw4_1_1 < ./test_case/12000-1.txt > ./test_case/myans_1.txt
[c0717001@linux1 hw_4]$ diff ./test_case/ans.txt ./test_case/myans_1.txt
1,3c1,3
< 0: 4044
< 1: 3973
< 2: 3983
---
> 0: 2945
> 1: 2918
> 2: 2878
[c0717001@linux1 hw_4]$
```

上半部分 diff 出來沒有錯的是有 synchronization，也就是有加入 semaphore 的版本；反觀下半部分，diff 有出錯的就是我把 semaphore 都註解掉以後的結果，可以看到因為 data race 的關係，所以會出錯。

Q2:
Compare results between hw4_1_2 with/without synchronization.

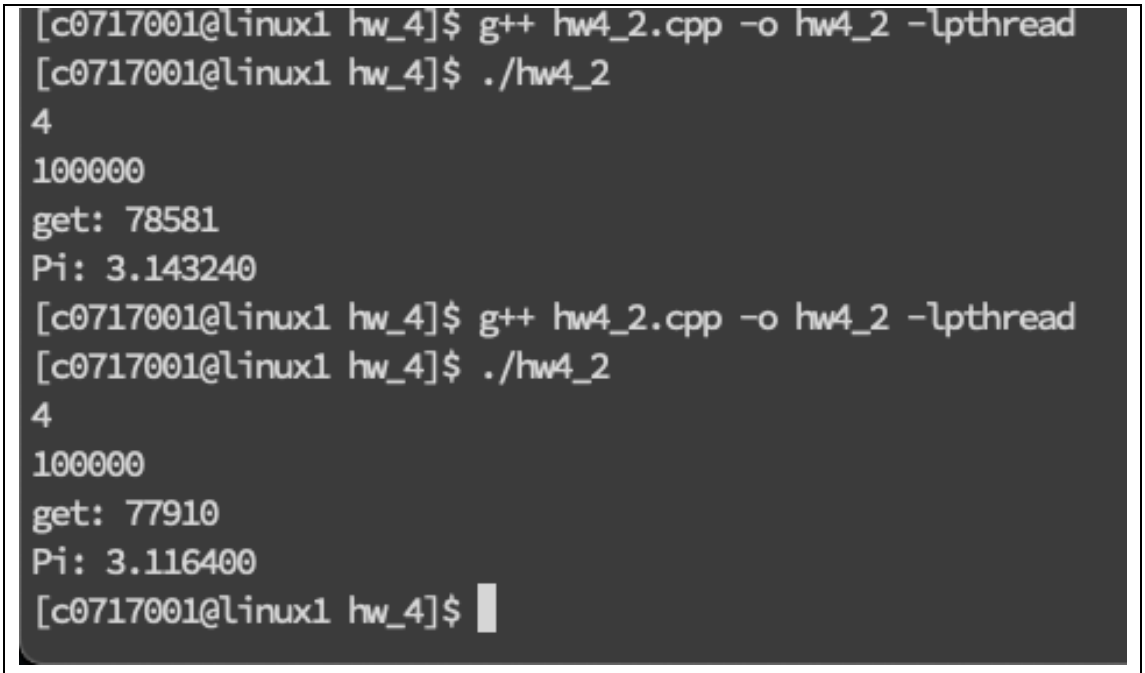
```
[c0717001@linux1 hw_4]$ g++ hw4_1_2.cpp -o hw4_1_2 -lpthread
[c0717001@linux1 hw_4]$ ./hw4_1_2 < ./test_case/12000-2.txt > ./test_case/myans_2.txt
[c0717001@linux1 hw_4]$ diff ./test_case/ans.txt ./test_case/myans_2.txt
[c0717001@linux1 hw_4]$ g++ hw4_1_2.cpp -o hw4_1_2 -lpthread
[c0717001@linux1 hw_4]$ ./hw4_1_2 < ./test_case/12000-2.txt > ./test_case/myans_2.txt
[c0717001@linux1 hw_4]$ diff ./test_case/ans.txt ./test_case/myans_2.txt
1,3c1,3
< 0: 4044
< 1: 3973
< 2: 3983
---
> 0: 2643
> 1: 2611
> 2: 2599
[c0717001@linux1 hw_4]$
```

上半部分 diff 出來沒有錯的是有 synchronization，也就是有加入 semaphore 的版本；反觀下半部分，diff 有出錯的就是我把 semaphore 都註

解掉以後的結果，可以看到因為 data race 的關係，所以會出錯。

Q3:

Compare results between hw4_2 with/without synchronization.



因為有 synchronization 可以有效阻止 data race 的狀況發生，所以算出來的答案也就比較接近真實的圓周率 3.14，反觀把 semaphore 註解掉的版本，也就是下半部分，圓周率的數值就比較不接近。

Q4:

Some problems you meet and how to resolve.

or some Reflections.

Semaphore 的用法研究了一下才會用，雖然這次作業好像可以不用 structure 包起來，但是上一次作業寫的記憶猶新，不知不覺就又用了 struct 去寫，後來問同學才知道好像根本不需要……總之還是挺好玩的一次作業，尤其是最後一題，我之前在一個國外的 youtuber 影片中有看到他介紹這個演算法，那時候才在驚嘆這個演算法也太神奇，居然用這種方式求得圓周率！沒有想到今天就被我給遇到，甚至還是作業系統的作業。寫起來感覺特別不一樣！

有卡比較久的地方應該是 4_1_2 題，他最後規定不能 cout 在 main 裏面，所以必須把 cout 寫在 void function 裏面，就要想辦法用迴圈 lock 控制 thread0 要比 thread1 先輸出，然後 thread1 要比 thread2 先輸出。一開始傻傻的把 cout 寫在 mainp program 裡面想說這第二小題到底在幹嘛……後來才意識到是自己的問題。

這學期在這堂課也學到很多，雖然期中考跟期末考沒有考到很高很理想，但是總的來說，還是獲益良多的一堂課，很喜歡老師的上課方式跟助教指派作業的仔細程度，辛苦了。