4.22

1.在 Linux 環境下(工作站)，使用指令 gcc –pthread –o target target.c

而後執行(以下均為./OS)，之後會要求一個 input，為 random 點的數量。

2. Result

```
cs60[~/os_homework3]-u103062129- gcc -pthread -o OS 4.22.c
cs60[~/os_homework3]-u103062129- ./OS
Monte Carlo method for calculating PI (multi thread)
Amount of random Points :
666
The approximating value of pi for the amount of points (666) is: 3.063063

cs60[~/os_homework3]-u103062129- ./OS
Monte Carlo method for calculating PI (multi thread)
Amount of random Points :
6666
The approximating value of pi for the amount of points (6666) is: 3.149115

cs60[~/os_homework3]-u103062129- ./OS
Monte Carlo method for calculating PI (multi thread)
Amount of random Points :
666666
The approximating value of pi for the amount of points (666666) is: 3.140025
```

3.

No race condition，since there's only single thread.

4.

作法說明：在 parent thread 另外建一條 child thread 來計算 random point，當 child thread 結束，由 parent thread 將存在 global variable(total points)中的值取出，以估算出 PI

4.23

1. 在 Linux 環境下(工作站)，使用指令 gcc –pthread –o target target.c

而後執行(以下均為./OS)，之後會要求一個 input，為 random 點的數量。

2. Result

```
cs60[~/os_homework3]-u103062129- gcc -pthread -o OS 4.23.c
cs60[~/os_homework3]-u103062129- ./OS
Monte Carlo method for calculating PI(openmp)
Amount of random Points :
666
The approximating value of pi for the amount of points (666) is: 3.183183

cs60[~/os_homework3]-u103062129- ./OS
Monte Carlo method for calculating PI(openmp)
Amount of random Points :
6666
The approximating value of pi for the amount of points (6666) is: 3.129313

cs60[~/os_homework3]-u103062129- ./OS
Monte Carlo method for calculating PI(openmp)
Amount of random Points :
666666
The approximating value of pi for the amount of points (666666) is: 3.144711
```

3.

因為 random point 的值為全域變數，因此有可能在 multi-thread 運算並寫入的時候發生 race condition。

4. 以 openMP 為架構，運用#pragma omp parallel 與 omp for，基本上概念與上相同。需要有一個 input，根據 Monte Carlo 的方法，輸入的值越大，則理論上出來的 PI 會越準確。

5.39

1. 在 Linux 環境下(工作站)，使用指令 gcc –pthread –o target target.c

而後執行(以下均為./OS)，之後會要求一個 input，為 random 點的數量。

2. Result

```
cs60[~/os_homework3]-u103062129- gcc -pthread -o OS 5.39.c
cs60[~/os_homework3]-u103062129- ./OS
Monte Carlo method for calculating PI(mutex lock)
Amount of random Points :
666
The approximating value of pi for the amount of points (666) is: 3.189189

cs60[~/os_homework3]-u103062129- ./OS
Monte Carlo method for calculating PI(mutex lock)
Amount of random Points :
66666
The approximating value of pi for the amount of points (66666) is: 3.142261

cs60[~/os_homework3]-u103062129- ./OS
Monte Carlo method for calculating PI(mutex lock)
Amount of random Points :
6666666
The approximating value of pi for the amount of points (6666666) is: 3.142345
```

3.

因為有使用 mutex lock 保護共享區(即 total points)，所以不會被同時存取，不會發生 race condition

4. 利用 mutex lock 來保護，一次僅會有一個 thread 可以拿到執行權，其他的 thread 會被 lock 住。

5.40

1. 在 Linux 環境下(工作站)，使用指令 gcc –pthread –o target target.c

而後執行(以下均為./OS)，之後會要求一個 input，為 random 點的數量。

2.

```
cs60[~/os_homework3]-u103062129- gcc -pthread -o OS 5.40.c
cs60[~/os_homework3]-u103062129- ./OS
Monte Carlo method for calculating PI(openmp with critical section)
Amount of random Points :
666
The approximating value of pi for the amount of points (666) is: 3.165165

cs60[~/os_homework3]-u103062129- ./OS
Monte Carlo method for calculating PI(openmp with critical section)
Amount of random Points :
66666
The approximating value of pi for the amount of points (66666) is: 3.142831

cs60[~/os_homework3]-u103062129- ./OS
Monte Carlo method for calculating PI(openmp with critical section)
Amount of random Points :
6666666
The approximating value of pi for the amount of points (6666666) is: 3.141900
```

3. 此解法不會有 race condition，4.22 中的 race condition 源自於全域變數，同時多條 thread 都有可能同時讀取或寫入 variable(totalPoints)。

4. 解法為， 利用 openMP 中的 #pragma omp critical，來控制保護共享的變數(totalPoints)，使得每一次只會有一個 thread 能夠 access。由結果可見，數值已經非常接近 PI，準確度也上升許多(3.1419)。