# Problem descriptions

在這次作業中，請使用 OpenMP 撰寫一個程式，該程式需要宣告 4 個Threads，並使用 for loop執行 1000000 次以計算 pi。這次有 4 個需要實作的programs，每個必須按照以下要求完成：

1. 使用 parallel 指令和 critical section 指令
2. 使用 parallel for 指令和 critical 指令
3. 移除第二題的 critical 指令，並觀察結果是否有錯誤
4. 使用第三題的程式碼，並結合 reduction 子句來計算出正確的 pi

# Code and explanations

## PPD\_PA2\_B0829002\_1.c

A computer screen shot of a program code

Description automatically generated

### Import library 和變數宣告

這部分用於印入C的Standard library和本次作業用到最重要的OpenMP library，然後Main function會把使用者執行程式後面連帶的arguments引入讓MPI在initialize parallel program的時候可以使用到。globalSum是用來儲存全域的summary、N是sigma的次數、THREAD是threads數目

A computer screen shot of a code

Description automatically generated

### 設定和初始化Parallel Program

設定thread數目後計算開始平行的時間，然後進入平行的部分就會先計算每個thread的頭尾，然後每個thread用算出來的頭尾跑for loop計算老師給的公式，之後就用critical section加入globalSum避免有mutual excecution，之後印出每個thread的結果。A white paper with black text

Description automatically generated

A computer screen shot of a computer code

Description automatically generated

### 結束時間以及印出結果

計算parallel的結束時間以及印出作業requirement的結果

A computer screen shot of a program code

Description automatically generated

## PPD\_PA2\_B0829002\_2.c

A computer screen shot of a program code

Description automatically generated

### Import library 、變數宣告和初始化Parallel Program

同PPD\_PA2\_B0829002\_1.c [[按我到之前的解釋]](#_Import_library_和變數宣告)

### 設定和初始化Parallel Program

這邊和前面的方法不太一樣，因為要用到pragma omp parallel for去處理parallel的部分所以要改下面這一小部分

A screen shot of a computer code

Description automatically generated

### 結束時間以及印出結果

計算parallel的結束時間以及印出作業requirement的結果

同PPD\_PA2\_B0829002\_1.c [[按我到之前的解釋]](#_結束時間以及印出結果)

## PPD\_PA2\_B0829002\_3.c

### Import library 、變數宣告和初始化Parallel Program

同PPD\_PA1\_B0829002\_1.c [[按我到之前的解釋]](#_Import_library_和變數宣告)

### Master Parallel Program

在這裡延續第二題的部分但要把critical section刪除來觀察是否會跑出錯誤結果

A computer screen shot of a program

Description automatically generated

### 結束時間以及印出結果

計算parallel的結束時間以及印出作業requirement的結果

同PPD\_PA2\_B0829002\_1.c [[按我到之前的解釋]](#_結束時間以及印出結果)

## PPD\_PA4\_B0829002\_3.c

A screen shot of a computer program

Description automatically generated

### Import library 、變數宣告和初始化Parallel Program

同PPD\_PA1\_B0829002\_1.c [[按我到之前的解釋]](#_Import_library_和變數宣告)

### Master Parallel Program

這裡大致上和前三題一樣但要base on No.3加上用reduction把globalSum加起來。

A screen shot of a computer code

Description automatically generated

### 結束時間以及印出結果

計算parallel的結束時間以及印出作業requirement的結果

同PPD\_PA2\_B0829002\_1.c [[按我到之前的解釋]](#_結束時間以及印出結果)

# Sampled outputs

## PPD\_PA2\_B0829002\_1.c

A computer screen shot of a program

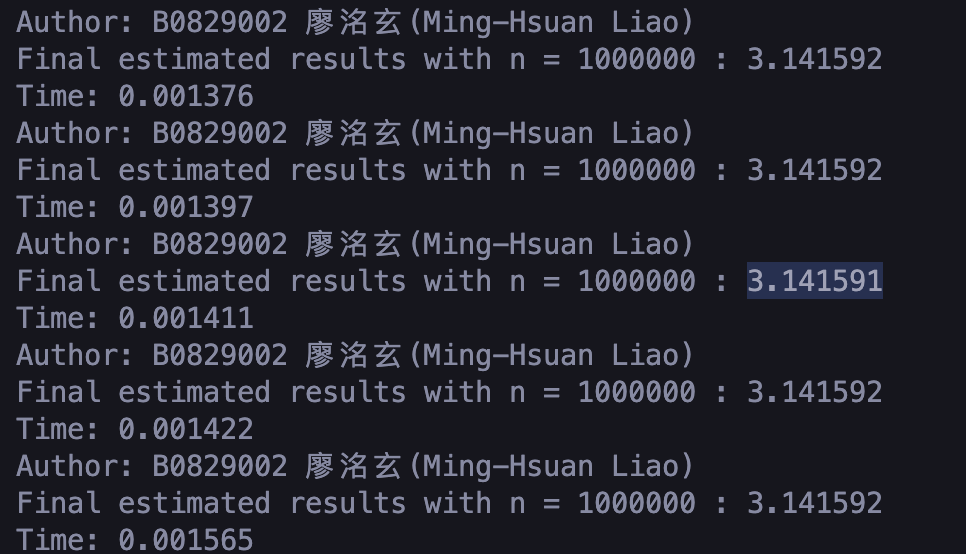
Description automatically generated

## PPD\_PA2\_B0829002\_2.c

A screen shot of a computer

Description automatically generated

## PPD\_PA2\_B0829002\_3.c



## PPD\_PA2\_B0829002\_4.c

A screen shot of a computer program

Description automatically generated

# Bonus

1. Recording the execution time of each program

從執行結果可以看到每個程式的執行時間（including Linux command: time and OpenMP time API: omp\_get\_wtime()）

1. Recording the speedup in problem 4 over the serial version by varying the number of threads = 1, 2, 4, 8,and 16

|  |  |  |
| --- | --- | --- |
| Number of threads | Time expenditure | Speedup Factor |
| 1 | 0.004335 | 1 |
| 2 | 0.004146 | 1.045586107 |
| 4 | 0.002556 | 1.69600939 |
| 8 | 0.001797 | 2.412353923 |
| 16 | 0.001592 | 2.72298995 |
| 32 | 0.002591 | 1.67309919 |
| 64 | 0.004951 | 1.142099193 |

1. Recording the approximated it global sum of problem 4 by varying n = 1000, 10000, 100000, and 1000000

可以看出來N越大PI值越準

|  |  |
| --- | --- |
| N | Estimation |
| 1000 | 3.140593 |
| 10000 | 3.141293 |
| 100000 | 3.141583 |
| 100000 | 3.141592 |
| 1000000 | 3.141593 |

# Discussions or what I’ve learned

在這個Assignment裡面我學習到了如何利用OpenMP提供的API開muti-thread，讓我的Program可以平行運算。而且我一開始原本想用自己的coding style寫

A black rectangle with white text

Description automatically generated

但發現compiler在編譯的時候會直接整段報錯，所以後來才改為omp.h定義好到coding style。除此之外，也深刻體會到使用 threads 進行並行處理所帶來的計算速度提升。與 process 並行相比，這四個programs在實作上並沒有太大的困難，只要好好聽課和看講義就會了。

# Reference

[1] L. Dagum and R. Menon, "OpenMP: an industry standard API for shared-memory programming," in IEEE Computational Science and Engineering, vol. 5, no. 1, pp. 46-55, Jan.-March 1998, doi: 10.1109/99.660313.keywords: {Message passing;Scalability;Hardware;Computer architecture;Power system modeling;ANSI standards;Parallel processing;Coherence;Software systems;Parallel programming},