OS HW2
Multi-Threading Programming

Operating System 111 Fall
Professor: W.J. TSAI

TA. 王菱君 王麗婷 黃逸弘 余孟倫

APIs

- **Thread management: <pthread.h>**
  - pthread_create
  - pthread_join
  - pthread_exit
- **Reference:**

[POSIX Threads Programming | LLNL HPC Tutorials]

Exercise - Hello Thread

```cpp
#include <iostream>
#include <thread>
#include <unistd.h>
using namespace std;

// child threading function
void* child_thread(void* data){
    sleep(1);
    cout << "Child Pthread ID - " << pthread_self() << endl;
    char *str = (char*) data; // get data "Child"
    for(int i = 0 ; i < 3 ; i++){
        sleep(1);
        cout << str << endl;// output every second
    }
    pthread_exit(NULL); // exit child thread
}

// main fuction
int main(void){
    pthread_t t;    // define thread
    pthread_create(&t,NULL,child_thread,(void *)"Child");   // create a child thread
    sleep(1);
    cout << "Master Pthread ID - " << pthread_self() << endl;   // output master thread's ID
    // pthread_join(t,NULL);   // wait for child threading finished, then output "Master"
    for(int i = 0 ; i < 3 ; i++){
        sleep(1);
        cout << "Master" << endl;

    }
    // pthread_join(t,NULL);// output "Master" during child thread outputing "Child"
    return 0;
}
```

<- hello_thread.cpp

```
$  g++ hello_thread.cpp -lpthread -o hello_thread
$ ./hello_thread
```

output:

```
Child Pthread ID - 0x16b93f000
Master Pthread ID - 0x104814580
Master
Child
Master
Child
Master
Child
```

Exercise - Single-threading

```cpp
# include <iostream>
# include <thread>
# include <unistd.h>
using namespace std;

# define MAX 500

int matA[MAX][MAX];
int matB[MAX][MAX];
int matC[MAX][MAX]; // Result of Addition
int matD[MAX][MAX]; // Result of Multiplication

void Addition(){
    // Addition -> matC
    for (int i = 0; i < MAX; i++) {
        for (int j = 0; j < MAX; j++){
            matC[i][j] = matA[i][j] + matB[i][j];
        }
    }
}
void Multiplication(){
    // Multiplication -> matD
    for (int i = 0; i < MAX; i++) {
        for (int j = 0; j < MAX; j++) {
            matD[i][j] = 0;
            for (int k = 0; k < MAX; k++) {
                matD[i][j] += matA[i][k] * matB[k][j];
            }
        }
    }
}

int main()
```

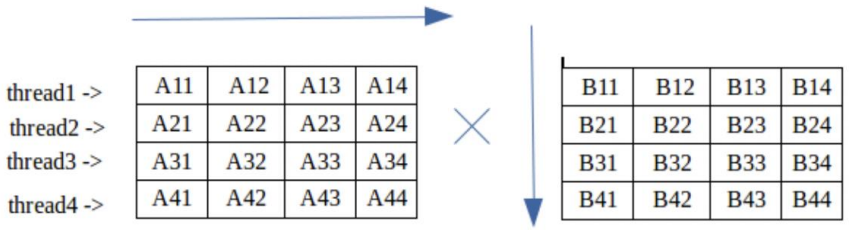
```
$  g++ single_thread.cpp -lpthread -o
single_thread
$ ./hello_thread < input.txt
```

output:

```
2248968
2528950360
```

Homework - Multithreading

- Use multithreading to do matrix calculation problems
- Find the best thread quantity between 2 ~ 20
- Output the sum of every element of your matrices



You should implement:

1. STDIN(e.g. scanf, cin)
2. Multiplication/addition function
3. Thread management
4. STDOUT(e.g. printf, cout)

DO NOT USE FILE I/O !

Compile & Run Commands

- **Compile:**

  (single-thread) $ g++ -o filename filename.cpp

  (multi-thread) $ g++ -o filename filename.cpp -lpthread

- **Run:**

  $ ./filename < input.txt

- **Environment:**

  You should run your code on the multiple CPU

Compile & Run Commands

- **Performance between single-thread and multi-thread:**

  use **“$ time ./filename”** to check the execution time

```
(base) jamie@jamie-System-Product-Name:~/Documents$ time ./ST >STout.txt

real    0m0.721s
user    0m0.709s
sys     0m0.012s
```

```
(base) jamie@jamie-System-Product-Name:~/Documents$ time ./MT1 >MT1out.txt

real    0m0.335s
user    0m1.074s
sys     0m0.047s
```

Speed up

- Compare the real time between single thread and the multi thread

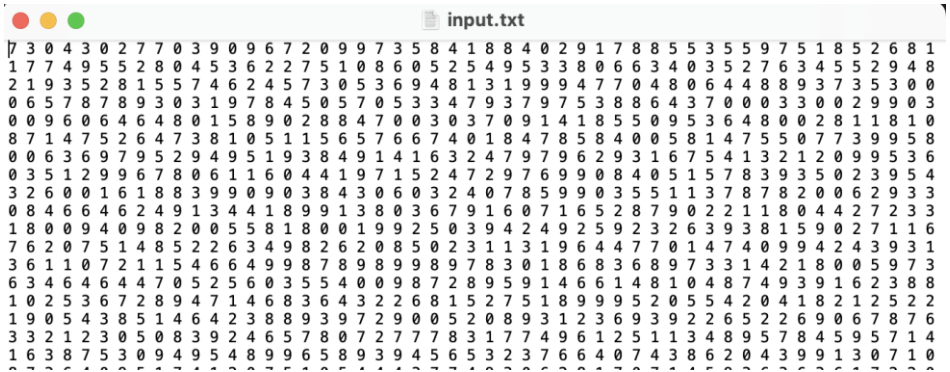
  Speedup =

  real time of multiple threads / real time of single thread

- Compare with your classmates (bigger speed-up gets higher score)
  - Top 1/5 get 20 points, second 1/5 get 16 points, etc.

Input format

- Input size : 500 * 500 matrix
- All elements are seperated by space (without newline)
- Matrix A start from line1, **matrix B start from line 2**
- Integer value random from 0~9



Output format

- Output the sum of every element of the Addition matrix **FIRST**
- Then output the sum of the Multiplication matrix **NEXT LINE**

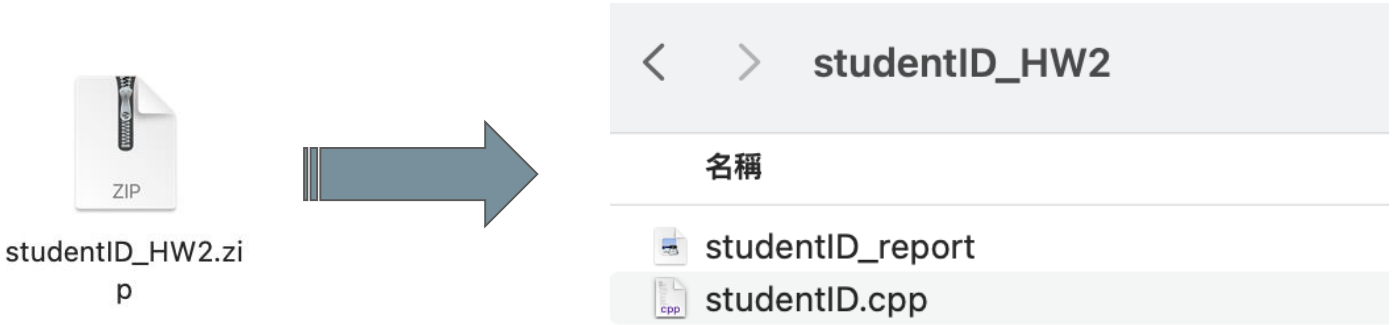
(Hint: If the sum is too big, try use long long int)

Requirements

- Multi-threading method should be much faster than Single-threading, and their results must be exactly the same.
- Write yout codes in C/C++
- You need to hand in one multi_thread versions and a report. Put studentID.cpp and studentID_report.pdf into the same compressed file without input, output data or any folder
- The type of compressed file **must** be “studentID_HW2.zip”
- **Use Ubuntu or NYCU work station as your environment.**

Requirements

- The compressed file needs to be as follow:



Grading

- Total score : 100 pts. **COPY WILL GET 0 POINT!**
- Speedup : 20 pts
- Multi-thread programs : 50 pts (correctness)
- Report : 30 pts
- Incorrect file format : -10 pts
- Use FILE I/O : -5 pts
- Deadline : 2022/10/30 (SUN) 23:59
- **Late submission will get a -20% point per day.**