# OS HW3
# Multi-Threading Programming

Operating System 110 Fall
Professor: W.J. TSAI

TA. 張皓雲 姚淨云 林孟學 王彥珽

# APIs

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

  POSIX Threads Programming | LLNL HPC Tutorials

# Exercise - Hello Thread

## hello_thread.c

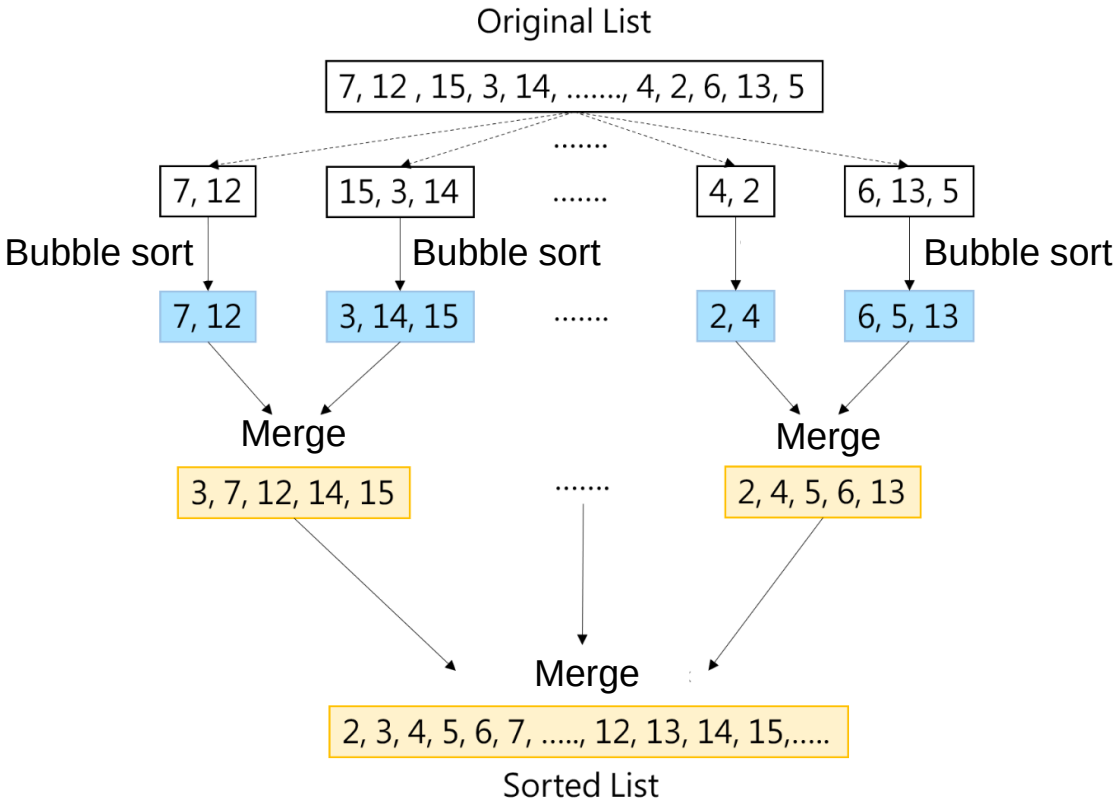
```c
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>

// child threading function
void* child(void* data){
    printf("Child Pthread ID - %lu\n", pthread_self());
    char *str = (char*) data; // get data "Child"
    int i;
    for(i=0;i<3;i++){
        // printf("%s\n", str);// output every second
        sleep(1);
    }
    pthread_exit(NULL); // exit child thread
}

// main fuction
int main(void){
    // define thread variable
    pthread_t t;
    // create child thread
    pthread_create(&t,NULL,child,(void *)"Child");
    printf("Mastetr Pthread ID - %lu\n", pthread_self());
    // main thread
    int i;
    for(i=0;i<3;i++){
        // printf("Master\n"); // output "Master" every second
        sleep(1);
    }
    pthread_join(t,NULL);// wait for all child threading finished
    return 0;
}
```

## Output

```
haoyuan@haoyuan-System-Product-Name:~/OS_HW3/OS_HW3$ g++ hello_thread.c -o hello_thread -lpthread
haoyuan@haoyuan-System-Product-Name:~/OS_HW3/OS_HW3$ ./hello_thread
Mastetr Pthread ID - 140226987423552
Child Pthread ID - 140226987419392
```

You don’t need to submit hello_thread.c and put screenshot in report.

# HW3-1 Single-threaded Sorting

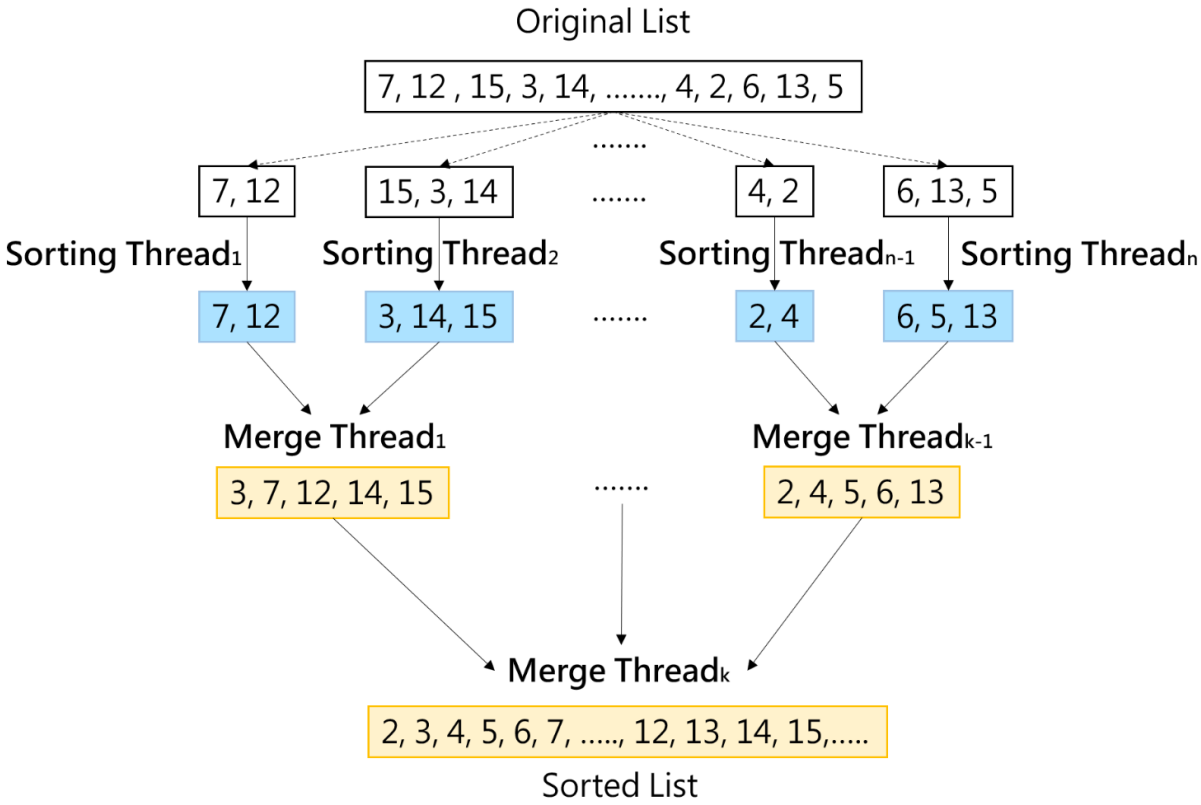


# HW3-1 Single-threaded Sorting

```c
#include <stdio.h>
#include <stdlib.h>

/* function definitions */
void *bubble_sort(void*);
void *merge(void*);

int main (int argc, const char * argv[])
{
    /* Use STDIN (e.g. scanf, cin) to take the input */
    /*
        your code here
    */

    /* Do the sorting */

    /* Use STDOUT (e.g. printf, cout) to output the sorted array */
    /*
        your code here
    */

    return 0;
}

void *bubble_sort(void *input_tinfo){



}

void *merge(void *input_tinfo){



}
```

- You should implement:
  1. STDIN (e.g. scanf, cin)
  2. buble_sort function (use brute-force methods)
  3. merge function
  4. STDOUT (e.g. printf, cout)
- **DO NOT USE FILE I/O !**
- **Single thread's partitions should be the same to multithread's partitions.**

# HW3-2 Multithreaded Sorting



# Multithreaded Sorting

- A list of integers is divided into several smaller lists.
- Using different threads (**sorting thread$_1$ ~ sorting thread$_n$**) sort each sublist using any brute-force methods (e.g., bubble sort).
- The several sublists are then merged into a single sorted list by threads (**merging thread$_1$ ~ merging thread$_k$.**
- You should implement:
    1. **STDIN** (e.g. scanf, cin)
    2. **Sorting thread function** (Use brute-force methods, e.g., bubble sort)
    3. **Merge thread function** (Use simple merge sort for merging two sublists)
    4. **Thread management**
    5. **STDOUT** (e.g. printf, cout)
- **DO NOT USE FILE I/O !**

# Compile & Run Commands

- **Compile:**

  (single-thread) $ g++ -o studentID_ST studentID_ST.c

  (multi-thread) $ g++ -o studentID_MT studentID_MT.c -lpthread

- **Run:**

  (single-thread) $ time ./studentID_ST < input1.txt > output1_ST.txt

  (multi-thread) $ time ./studentID_MT < input1.txt > output1_MT.txt

- **Environment**
  - You should run your code on the multiple CPU.

# Compile & Run Commands

- Performance between single thread and multi thread:

```
haoyuan@haoyuan-System-Product-Name:~/OS_HW3/0816000_hw3$ g++ 0816000_MT.cpp -o 0816000_MT -lpthread
haoyuan@haoyuan-System-Product-Name:~/OS_HW3/0816000_hw3$ time ./0816000_MT < input.txt > output.txt

real    0m0.065s
user    0m0.168s
sys     0m0.004s
```

```
haoyuan@haoyuan-System-Product-Name:~/OS_HW3/0816000_hw3$ g++ 0816000_ST.cpp -o 0816000_ST -lpthread
haoyuan@haoyuan-System-Product-Name:~/OS_HW3/0816000_hw3$ time ./0816000_ST < input.txt > output.txt

real    0m0.285s
user    0m0.276s
sys     0m0.008s
```

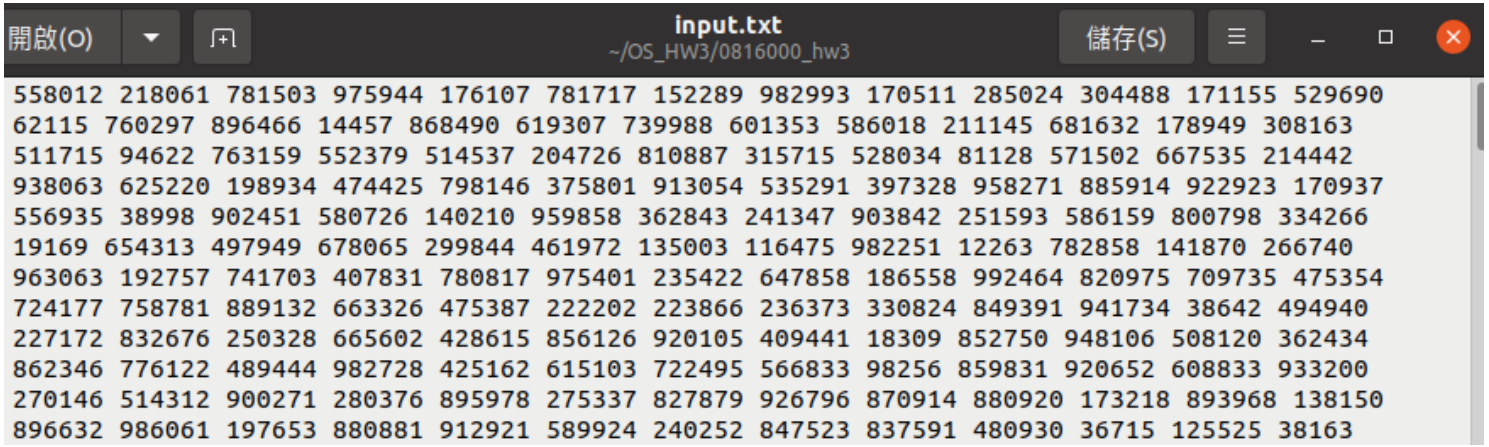
- Find the difference between answer and your output:

  diff --suppress-common-lines answer{1,2}.txt ouput{1, 2}.txt

  (answer{1, 2}.txt is provided by TA)

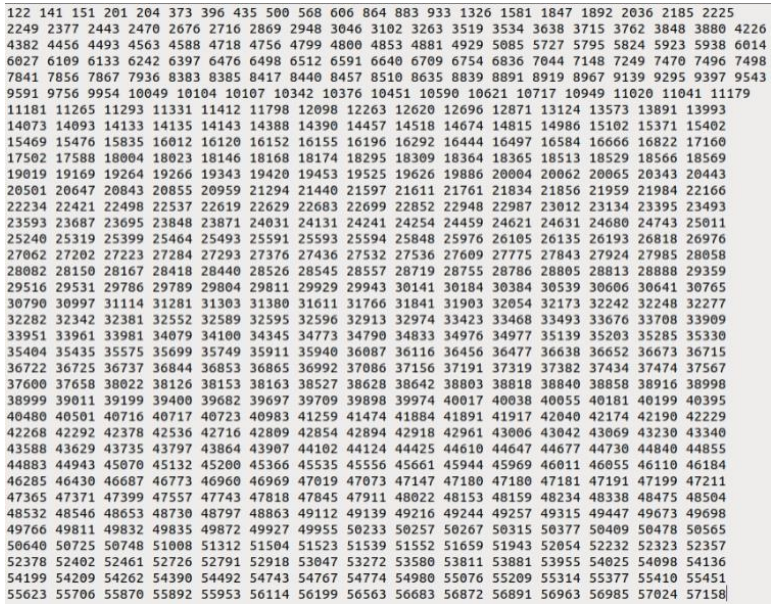
# Input/output format

- Input format:
  - All elements will separated by space
  - Largest input: 1,000,000 integers
  - Integer value was random from 0 to 1000000



# Input/output format

- Output format:



```
122 141 151 201 204 373 396 435 500 568 606 864 883 933 1326 1581 1847 1892 2036 2185 2225
2249 2377 2443 2470 2676 2716 2869 2948 3046 3102 3263 3519 3534 3638 3715 3762 3848 3880 4226
4382 4456 4493 4563 4588 4718 4756 4799 4800 4853 4881 4929 5085 5727 5795 5824 5923 5938 6014
6027 6109 6133 6242 6397 6476 6498 6512 6591 6640 6709 6754 6836 7044 7148 7249 7470 7496 7498
7841 7856 7867 7936 8383 8385 8417 8440 8457 8510 8635 8839 8891 8919 8967 9139 9295 9397 9543
9591 9756 9954 10049 10104 10107 10342 10376 10451 10590 10621 10717 10949 11020 11041 11179
11181 11265 11293 11331 11412 11798 12098 12263 12620 12696 12871 13124 13573 13891 13993
14073 14093 14133 14135 14143 14388 14390 14457 14518 14674 14815 14986 15102 15371 15402
15469 15476 15835 16012 16120 16152 16155 16196 16292 16444 16497 16584 16666 16822 17160
17502 17588 18004 18023 18146 18168 18174 18295 18309 18364 18365 18513 18529 18566 18569
19019 19169 19264 19266 19343 19420 19453 19525 19626 19886 20004 20062 20065 20343 20443
20501 20647 20843 20855 20959 21294 21440 21597 21611 21761 21834 21856 21959 21984 22166
22234 22421 22498 22537 22619 22629 22683 22699 22852 22948 22987 23012 23134 23395 23493
23593 23687 23695 23848 23871 24031 24131 24241 24254 24459 24621 24631 24680 24743 25011
25240 25319 25399 25464 25493 25591 25593 25594 25848 25976 26105 26135 26193 26818 26976
27062 27202 27223 27284 27293 27376 27436 27532 27536 27609 27775 27843 27924 27985 28058
28082 28150 28167 28418 28440 28526 28545 28557 28719 28755 28786 28805 28813 28888 29359
29516 29531 29786 29789 29804 29811 29929 29943 30141 30184 30384 30539 30606 30641 30765
30790 30997 31114 31281 31303 31380 31611 31766 31841 31903 32054 32173 32242 32248 32277
32282 32342 32381 32552 32589 32595 32596 32913 32974 33423 33468 33493 33676 33708 33909
33951 33961 33981 34079 34100 34345 34773 34790 34833 34976 34977 35139 35203 35285 35330
35404 35435 35575 35699 35749 35911 35940 36087 36116 36456 36477 36638 36652 36673 36715
36722 36725 36737 36844 36853 36865 36992 37086 37156 37191 37319 37382 37434 37474 37567
37600 37658 38022 38126 38153 38163 38527 38628 38642 38803 38818 38840 38858 38916 38998
38999 39011 39199 39400 39682 39697 39709 39898 39974 40017 40038 40055 40181 40199 40395
40480 40501 40716 40717 40723 40983 41259 41474 41884 41891 41917 42040 42174 42190 42229
42268 42292 42378 42536 42716 42809 42854 42894 42918 42961 43006 43042 43069 43230 43340
43588 43629 43735 43797 43864 43907 44102 44124 44425 44610 44647 44677 44730 44840 44855
44883 44943 45070 45132 45200 45366 45535 45556 45661 45944 45969 46011 46055 46110 46184
46285 46430 46687 46773 46960 46969 47019 47073 47147 47180 47180 47181 47191 47199 47211
47365 47371 47399 47557 47743 47818 47845 47911 48022 48153 48159 48234 48338 48475 48504
48532 48546 48653 48730 48797 48863 49112 49139 49216 49244 49257 49315 49447 49673 49698
49766 49811 49832 49835 49872 49927 49955 50233 50257 50267 50315 50377 50409 50478 50565
50640 50725 50748 51008 51312 51504 51523 51539 51552 51659 51943 52054 52232 52323 52357
52378 52402 52461 52726 52791 52918 53047 53272 53580 53811 53881 53955 54025 54098 54136
54199 54209 54262 54390 54492 54743 54767 54774 54980 55076 55209 55314 55377 55410 55451
55623 55706 55870 55892 55953 56114 56199 56563 56683 56872 56891 56963 56985 57024 57158
```

output_{ST, MT}_{1, 2}.txt

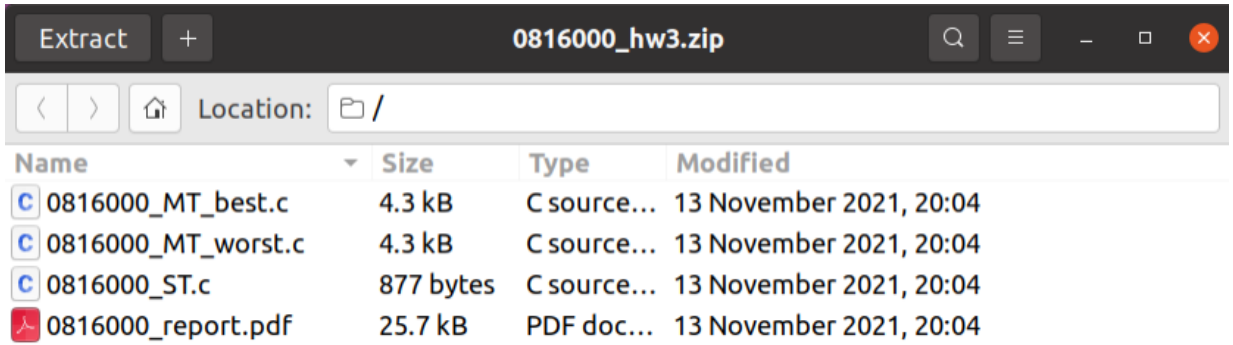
1. Use SPACE between every two numbers
2. No need newline

# Requirements

- The sorting threads should use **the same** sorting algorithm as the single-thread program.
- Multi-thread sorting should be much faster than Single-thread sorting, and their results must be exactly the same.
- Write your codes in **c/c++**
- You need to hand in one single-thread version and the other multithread version. Put **studentID_ST.c(.cpp), studentID_MT_best.c(.cpp), studentID_MT_worst.c(.cpp) and studentID_report.pdf** into the same compressed file without folder.
- The type of compressed file must be "**studentID_hw3.zip**"
- **Use Ubuntu or NYCU work station as your environment.**

# Requirements

- The compressed file needs to be compressed as follows:



# Grading

- Total score: 100 pts. **COPY WILL GET 0 POINT!**
- Single-thread program: 20 pts (correctness)
- Two Multi-thread programs: 40 pts (correctness)
- Report: 40 pts
- Incorrect file format: -10 pts
- Use FILE I/O: -5 pts
- Deadline: **2021/12/1 (Wed) 23:59**

**Late submission will get a -20% point per day.**