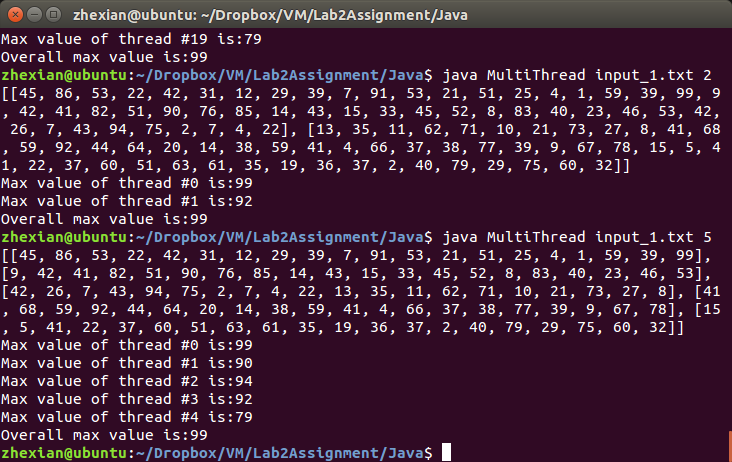
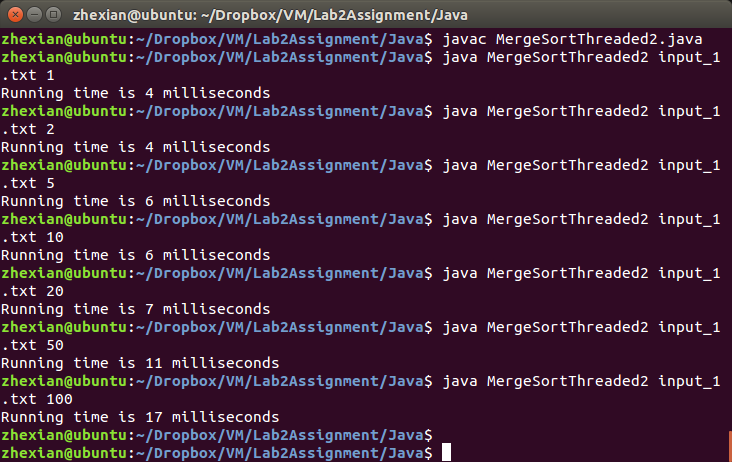
# Question 1:



# Question 2:



**Execution time record\*:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Number of Threads | 1 | 2 | 5 | 10 | 20 | 50 | 100 |
| Execution Time (ms) | 4 | 4 | 6 | 6 | 7 | 11 | 17 |

(\*Note: the above execution time is derived by executing input\_1.txt, not input\_2.txt, because for the later the program did not return any result after an hour. This could due to the performance limitation of my dual-core processor, and the single processor allocated to the Linux virtual machine. In order to still get some result for analysis I choose to compare the execution time of running input\_1.txt instead.)

**Analysis on the relationship between number of threads and execution time:**

Using multi-thread is supposed to speed up and improve performance for certain computing process, but it is not so in my case.

The main reason (and a common misconception) is that threads cannot speed up execution of code. They do not make the computer run faster, but instead increase the efficiency of the computer by using time that would otherwise be wasted – by fully utilizing CPU processing power for multitasking (running a few processes in parallel).

Threading is not always quicker and in many cases can be slower due to the following reasons:

1. Creating a thread is a relatively expensive OS operation
2. Context switching (where the CPU stops working on one thread and starts working on another) is also a relatively expensive operation, when the number of threads exceed the number of CPU cores.

The optimum result should be obtained when the number of thread is the same as number of cores my machine has, in which case each core can process one thread without being interrupted, and thus improve performance.

However, since the virtual machine for my Linux system only has one core, the performance is only optimized when one thread is used (i.e. sequential execution). The more thread I used, the more time it takes due to time wasted in creating threads and context switching.