Multicore computing

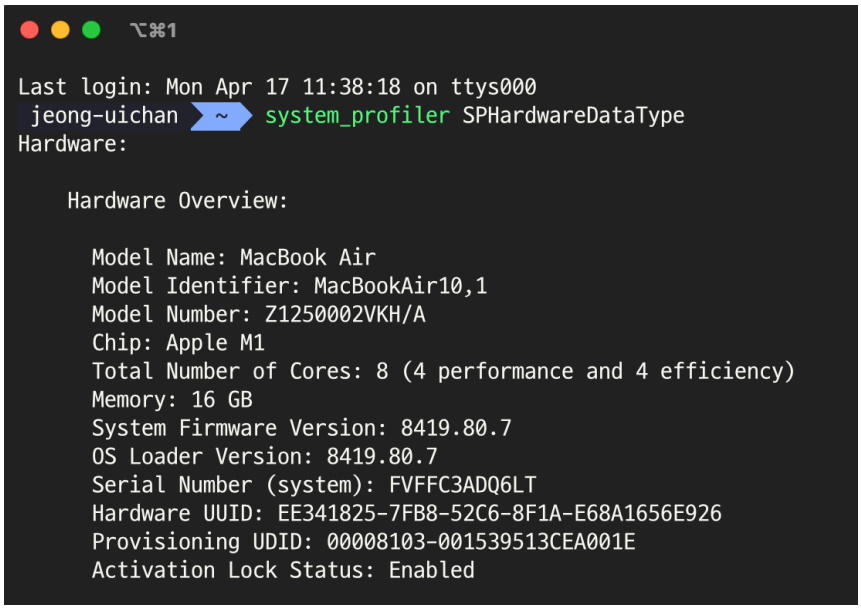
Project #1 – problem1

Department : Software

Name : Jeong Eui Chan

Student ID : 20195914

MY computer system enviromnent



Both Problem1 and Problem2 were performed in an environment with the following specifications:
The cpu of my computer uses an Apple M1 chip with 8 cores, consisting of 4 P-cores and 4 E-cores and The maximum clock speed of my cpu is 3.2GH.
The memory capacity is 16GB, and the memory type used is LPDDR4. The operating system being used is macOS Ventrura version 13.2.1.

1. Block decomposition

My Block decomposition code performs multi-threading to find prime numbers from 1 to 200,000 and measures the execution time of each thread and the total program time. The number of working threads is changed through the given Entire Threads, and the changing results are checked. Parallel processing is performed to find prime numbers according to the number of threads, and each thread works in a separate range. By doing this, prime numbers within the entire range can be found more quickly. Since it uses static load balancing with Block decomposition, the size of the block is adjusted, and threads are assigned to process the blocks. The code I wrote is added at the end.

1-1) How to compile & execute

To create the source code, I first created a project in Eclipse, then created a project file called "Thread test." Next, I created a package called "Problem1_block." I then created a class named "pc_stack_block" I wrote the code in this class, and to compile and execute it, I clicked on "Run." As a result, the program outputs the number of primes, the time taken by each thread, and the total program execution time, as shown above.

1-2) Result screen

[Entire Threads : 1]

```
Entire Threads : 1

(1) [execution time of each thread]
    Thread-1 Execution Time : 43ms

(2) [program execution time]
    Program Execution Time : 45ms

(3) [the number of prime numbers]
    Thread-1 Range: 0 to 199999, Prime Count: 17984

1 ... 199999 prime# counter = 17984

------------------------------
```

[Entire Threads : 2]

```
Entire Threads : 2

(1) [execution time of each thread]
    Thread-1 Execution Time : 21ms
    Thread-2 Execution Time : 25ms

(2) [program execution time]
    Program Execution Time : 26ms

(3) [the number of prime numbers]
    Thread-1 Range: 0 to 99999, Prime Count: 9592
    Thread-2 Range: 100000 to 199999, Prime Count: 8392

1 ... 199999 prime# counter = 17984

------------------------------
```

[Entire Threads : 4]

```
Entire Threads : 4

(1) [execution time of each thread]
    Thread-1 Execution Time : 3ms
    Thread-2 Execution Time : 4ms
    Thread-3 Execution Time : 5ms
    Thread-4 Execution Time : 7ms

(2) [program execution time]
    Program Execution Time : 7ms

(3) [the number of prime numbers]
    Thread-1 Range: 0 to 49999, Prime Count: 5133
    Thread-2 Range: 50000 to 99999, Prime Count: 4459
    Thread-3 Range: 100000 to 149999, Prime Count: 4256
    Thread-4 Range: 150000 to 199999, Prime Count: 4136

1 ... 199999 prime# counter = 17984

------------------------------
```

[Entire Threads : 8]

```
Entire Threads : 8

(1) [execution time of each thread]
    Thread-1 Execution Time : 3ms
    Thread-2 Execution Time : 2ms
    Thread-3 Execution Time : 2ms
    Thread-4 Execution Time : 3ms
    Thread-5 Execution Time : 5ms
    Thread-6 Execution Time : 3ms
    Thread-7 Execution Time : 3ms
    Thread-8 Execution Time : 4ms

(2) [program execution time]
    Program Execution Time : 7ms

(3) [the number of prime numbers]
    Thread-1 Range: 0 to 24999, Prime Count: 2762
    Thread-2 Range: 25000 to 49999, Prime Count: 2371
    Thread-3 Range: 50000 to 74999, Prime Count: 2260
    Thread-4 Range: 75000 to 99999, Prime Count: 2199
    Thread-5 Range: 100000 to 124999, Prime Count: 2142
    Thread-6 Range: 125000 to 149999, Prime Count: 2114
    Thread-7 Range: 150000 to 174999, Prime Count: 2068
    Thread-8 Range: 175000 to 199999, Prime Count: 2068

1 ... 199999 prime# counter = 17984

------------------------------
```

[Entire Threads : 10]

```
Entire Threads : 10

(1) [execution time of each thread]
    Thread-1 Execution Time : 1ms
    Thread-2 Execution Time : 2ms
    Thread-3 Execution Time : 4ms
    Thread-4 Execution Time : 2ms
    Thread-5 Execution Time : 4ms
    Thread-6 Execution Time : 6ms
    Thread-7 Execution Time : 6ms
    Thread-8 Execution Time : 4ms
    Thread-9 Execution Time : 3ms
    Thread-10 Execution Time : 4ms

(2) [program execution time]
    Program Execution Time : 7ms

(3) [the number of prime numbers]
    Thread-1 Range: 0 to 19999, Prime Count: 2262
    Thread-2 Range: 20000 to 39999, Prime Count: 1941
    Thread-3 Range: 40000 to 59999, Prime Count: 1854
    Thread-4 Range: 60000 to 79999, Prime Count: 1780
    Thread-5 Range: 80000 to 99999, Prime Count: 1755
    Thread-6 Range: 100000 to 119999, Prime Count: 1709
    Thread-7 Range: 120000 to 139999, Prime Count: 1709
    Thread-8 Range: 140000 to 159999, Prime Count: 1673
    Thread-9 Range: 160000 to 179999, Prime Count: 1659
    Thread-10 Range: 180000 to 199999, Prime Count: 1642

1 ... 199999 prime# counter = 17984

------------------------------
```

[Entire Threads : 12]

```
Entire Threads : 12

(1) [execution time of each thread]
    Thread-1 Execution Time : 0ms
    Thread-2 Execution Time : 2ms
    Thread-3 Execution Time : 1ms
    Thread-4 Execution Time : 1ms
    Thread-5 Execution Time : 2ms
    Thread-6 Execution Time : 3ms
    Thread-7 Execution Time : 2ms
    Thread-8 Execution Time : 2ms
    Thread-9 Execution Time : 3ms
    Thread-10 Execution Time : 3ms
    Thread-11 Execution Time : 3ms
    Thread-12 Execution Time : 3ms

(2) [program execution time]
    Program Execution Time : 6ms

(3) [the number of prime numbers]
    Thread-1 Range: 0 to 16665, Prime Count: 1928
    Thread-2 Range: 16666 to 33331, Prime Count: 1641
    Thread-3 Range: 33332 to 49997, Prime Count: 1563
    Thread-4 Range: 49998 to 66663, Prime Count: 1513
    Thread-5 Range: 66664 to 83329, Prime Count: 1489
    Thread-6 Range: 83330 to 99995, Prime Count: 1458
    Thread-7 Range: 99996 to 116661, Prime Count: 1422
    Thread-8 Range: 116662 to 133327, Prime Count: 1430
    Thread-9 Range: 133328 to 149993, Prime Count: 1404
    Thread-10 Range: 149994 to 166659, Prime Count: 1377
    Thread-11 Range: 166660 to 183325, Prime Count: 1381
    Thread-12 Range: 183326 to 199999, Prime Count: 1378

1 ... 199999 prime# counter = 17984

------------------------------
```

[Entire Threads : 14]

```
Entire Threads : 14

(1) [execution time of each thread]
    Thread-1 Execution Time : 1ms
    Thread-2 Execution Time : 1ms
    Thread-3 Execution Time : 1ms
    Thread-4 Execution Time : 1ms
    Thread-5 Execution Time : 2ms
    Thread-6 Execution Time : 3ms
    Thread-7 Execution Time : 3ms
    Thread-8 Execution Time : 2ms
    Thread-9 Execution Time : 2ms
    Thread-10 Execution Time : 2ms
    Thread-11 Execution Time : 2ms
    Thread-12 Execution Time : 3ms
    Thread-13 Execution Time : 2ms
    Thread-14 Execution Time : 2ms

(2) [program execution time]
    Program Execution Time : 7ms

(3) [the number of prime numbers]
    Thread-1 Range: 0 to 14284, Prime Count: 1676
    Thread-2 Range: 14285 to 28569, Prime Count: 1431
    Thread-3 Range: 28570 to 42854, Prime Count: 1374
    Thread-4 Range: 42855 to 57139, Prime Count: 1313
    Thread-5 Range: 57140 to 71424, Prime Count: 1280
    Thread-6 Range: 71425 to 85709, Prime Count: 1267
    Thread-7 Range: 85710 to 99994, Prime Count: 1251
    Thread-8 Range: 99995 to 114279, Prime Count: 1222
    Thread-9 Range: 114280 to 128564, Prime Count: 1224
    Thread-10 Range: 128565 to 142849, Prime Count: 1214
    Thread-11 Range: 142850 to 157134, Prime Count: 1194
    Thread-12 Range: 157135 to 171419, Prime Count: 1169
    Thread-13 Range: 171420 to 185704, Prime Count: 1191
    Thread-14 Range: 185705 to 199999, Prime Count: 1178

1 ... 199999 prime# counter = 17984

------------------------------
```

[Entire Threads : 16]

```
Entire Threads : 16

(1) [execution time of each thread]
    Thread-1 Execution Time : 1ms
    Thread-2 Execution Time : 1ms
    Thread-3 Execution Time : 1ms
    Thread-4 Execution Time : 2ms
    Thread-5 Execution Time : 1ms
    Thread-6 Execution Time : 1ms
    Thread-7 Execution Time : 2ms
    Thread-8 Execution Time : 1ms
    Thread-9 Execution Time : 2ms
    Thread-10 Execution Time : 2ms
    Thread-11 Execution Time : 3ms
    Thread-12 Execution Time : 2ms
    Thread-13 Execution Time : 2ms
    Thread-14 Execution Time : 2ms
    Thread-15 Execution Time : 2ms
    Thread-16 Execution Time : 2ms

(2) [program execution time]
    Program Execution Time : 6ms

(3) [the number of prime numbers]
    Thread-1 Range: 0 to 12499, Prime Count: 1492
    Thread-2 Range: 12500 to 24999, Prime Count: 1270
    Thread-3 Range: 25000 to 37499, Prime Count: 1206
    Thread-4 Range: 37500 to 49999, Prime Count: 1165
    Thread-5 Range: 50000 to 62499, Prime Count: 1142
    Thread-6 Range: 62500 to 74999, Prime Count: 1118
    Thread-7 Range: 75000 to 87499, Prime Count: 1099
    Thread-8 Range: 87500 to 99999, Prime Count: 1100
    Thread-9 Range: 100000 to 112499, Prime Count: 1070
    Thread-10 Range: 112500 to 124999, Prime Count: 1072
    Thread-11 Range: 125000 to 137499, Prime Count: 1068
    Thread-12 Range: 137500 to 149999, Prime Count: 1046
    Thread-13 Range: 150000 to 162499, Prime Count: 1037
    Thread-14 Range: 162500 to 174999, Prime Count: 1031
    Thread-15 Range: 175000 to 187499, Prime Count: 1048
    Thread-16 Range: 187500 to 199999, Prime Count: 1020

1 ... 199999 prime# counter = 17984
```

[Entire Threads : 32]

```
Entire Threads : 32

(1) [execution time of each thread]
    Thread-1 Execution Time : 0ms
    Thread-2 Execution Time : 1ms
    Thread-3 Execution Time : 0ms
    Thread-4 Execution Time : 0ms
    Thread-5 Execution Time : 0ms
    Thread-6 Execution Time : 0ms
    Thread-7 Execution Time : 1ms
    Thread-8 Execution Time : 1ms
    Thread-9 Execution Time : 1ms
    Thread-10 Execution Time : 1ms
    Thread-11 Execution Time : 1ms
    Thread-12 Execution Time : 1ms
    Thread-13 Execution Time : 0ms
    Thread-14 Execution Time : 1ms
    Thread-15 Execution Time : 1ms
    Thread-16 Execution Time : 1ms
    Thread-17 Execution Time : 0ms
    Thread-18 Execution Time : 1ms
    Thread-19 Execution Time : 1ms
    Thread-20 Execution Time : 1ms
    Thread-21 Execution Time : 1ms
    Thread-22 Execution Time : 1ms
    Thread-23 Execution Time : 0ms
    Thread-24 Execution Time : 2ms
    Thread-25 Execution Time : 1ms
    Thread-26 Execution Time : 1ms
    Thread-27 Execution Time : 1ms
    Thread-28 Execution Time : 1ms
    Thread-29 Execution Time : 1ms
    Thread-30 Execution Time : 1ms
    Thread-31 Execution Time : 2ms
    Thread-32 Execution Time : 1ms

(2) [program execution time]
    Program Execution Time : 8ms

(3) [the number of prime numbers]
    Thread-1 Range: 0 to 6249, Prime Count: 812
    Thread-2 Range: 6250 to 12499, Prime Count: 680
    Thread-3 Range: 12500 to 18749, Prime Count: 648
    Thread-4 Range: 18750 to 24999, Prime Count: 622
    Thread-5 Range: 25000 to 31249, Prime Count: 606
    Thread-6 Range: 31250 to 37499, Prime Count: 600
    Thread-7 Range: 37500 to 43749, Prime Count: 587
    Thread-8 Range: 43750 to 49999, Prime Count: 578
    Thread-9 Range: 50000 to 56249, Prime Count: 573
    Thread-10 Range: 56250 to 62499, Prime Count: 569
    Thread-11 Range: 62500 to 68749, Prime Count: 559
    Thread-12 Range: 68750 to 74999, Prime Count: 559
    Thread-13 Range: 75000 to 81249, Prime Count: 559
    Thread-14 Range: 81250 to 87499, Prime Count: 540
    Thread-15 Range: 87500 to 93749, Prime Count: 555
    Thread-16 Range: 93750 to 99999, Prime Count: 545
    Thread-17 Range: 100000 to 106249, Prime Count: 533
    Thread-18 Range: 106250 to 112499, Prime Count: 537
    Thread-19 Range: 112500 to 118749, Prime Count: 532
    Thread-20 Range: 118750 to 124999, Prime Count: 540
    Thread-21 Range: 125000 to 131249, Prime Count: 529
    Thread-22 Range: 131250 to 137499, Prime Count: 539
    Thread-23 Range: 137500 to 143749, Prime Count: 522
    Thread-24 Range: 143750 to 149999, Prime Count: 524
    Thread-25 Range: 150000 to 156249, Prime Count: 528
    Thread-26 Range: 156250 to 162499, Prime Count: 509
    Thread-27 Range: 162500 to 168749, Prime Count: 509
    Thread-28 Range: 168750 to 174999, Prime Count: 522
    Thread-29 Range: 175000 to 181249, Prime Count: 517
    Thread-30 Range: 181250 to 187499, Prime Count: 531
    Thread-31 Range: 187500 to 193749, Prime Count: 510
    Thread-32 Range: 193750 to 199999, Prime Count: 510

1 ... 199999 prime# counter = 17984

----------------------------
```

2. Cyclic decomposition

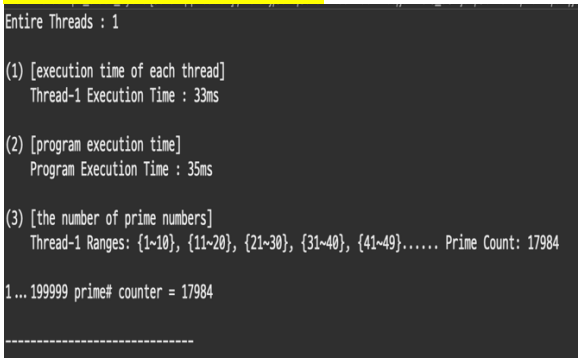
My Cyclic decomposition code uses multithreading to find prime numbers from 1 to 200,000, measuring the execution time of each thread and the total program time. The number of working threads is changed through the given entire threads, and the change results are checked. Parallel processing is performed to find prime numbers depending on the number of threads, and each thread operates in a separate range. Unlike Block decomposition, the entire workload is broken down into smaller units and assigned to threads. As the task size is explained as 10 in this assignment, the work units are decomposed into groups of 10 numbers. Through such work, the probability of a specific thread remaining idle while other threads are still processing tasks can be reduced.

2-1) How to compile & execute

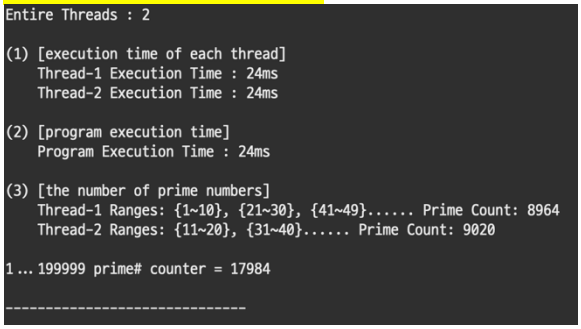
To create the source code, I first created a project in Eclipse, then created a project file called "Thread test." Next, I created a package called "Problem1_cyclic" I then created a class named "pc_stack_cyclic" I wrote the code in this class, and to compile and execute it, I clicked on "Run." As a result, the program outputs the number of primes, the time taken by each thread, and the total program execution time, as shown above.

2-2) Result screen

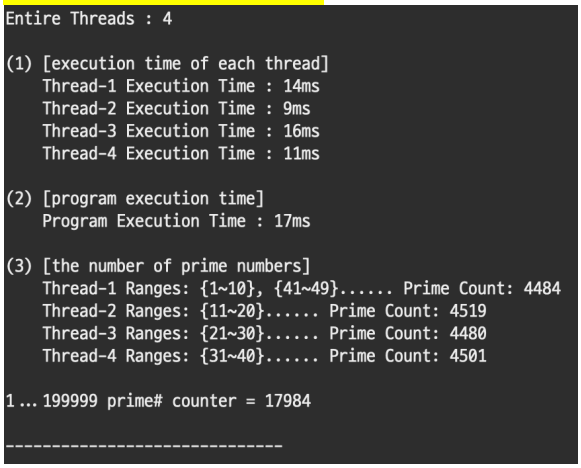
[Entire Threads : 1]



[Entire Threads : 2]



[Entire Threads : 4]



[Entire Threads : 6]

```
Entire Threads : 6

(1) [execution time of each thread]
    Thread-1 Execution Time : 3ms
    Thread-2 Execution Time : 6ms
    Thread-3 Execution Time : 4ms
    Thread-4 Execution Time : 3ms
    Thread-5 Execution Time : 6ms
    Thread-6 Execution Time : 4ms

(2) [program execution time]
    Program Execution Time : 8ms

(3) [the number of prime numbers]
    Thread-1 Ranges: {1~10}..... Prime Count: 2231
    Thread-2 Ranges: {11~20}..... Prime Count: 4507
    Thread-3 Ranges: {21~30}..... Prime Count: 2231
    Thread-4 Ranges: {31~40}..... Prime Count: 2252
    Thread-5 Ranges: {41~49}..... Prime Count: 4502
    Thread-6 Ranges: ..... Prime Count: 2261

1 ... 199999 prime# counter = 17984

-----------------------------
```

[Entire Threads : 8]

```
Entire Threads : 8

(1) [execution time of each thread]
    Thread-1 Execution Time : 3ms
    Thread-2 Execution Time : 5ms
    Thread-3 Execution Time : 5ms
    Thread-4 Execution Time : 5ms
    Thread-5 Execution Time : 6ms
    Thread-6 Execution Time : 3ms
    Thread-7 Execution Time : 5ms
    Thread-8 Execution Time : 7ms

(2) [program execution time]
    Program Execution Time : 8ms

(3) [the number of prime numbers]
    Thread-1 Ranges: {1~10}..... Prime Count: 2244
    Thread-2 Ranges: {11~20}..... Prime Count: 2251
    Thread-3 Ranges: {21~30}..... Prime Count: 2238
    Thread-4 Ranges: {31~40}..... Prime Count: 2246
    Thread-5 Ranges: {41~49}..... Prime Count: 2240
    Thread-6 Ranges: ..... Prime Count: 2268
    Thread-7 Ranges: ..... Prime Count: 2242
    Thread-8 Ranges: ..... Prime Count: 2255

1 ... 199999 prime# counter = 17984

-----------------------------
```

[Entire Threads : 10]

```
Entire Threads : 10

(1) [execution time of each thread]
    Thread-1 Execution Time : 4ms
    Thread-2 Execution Time : 3ms
    Thread-3 Execution Time : 3ms
    Thread-4 Execution Time : 3ms
    Thread-5 Execution Time : 2ms
    Thread-6 Execution Time : 3ms
    Thread-7 Execution Time : 2ms
    Thread-8 Execution Time : 2ms
    Thread-9 Execution Time : 2ms
    Thread-10 Execution Time : 2ms

(2) [program execution time]
    Program Execution Time : 6ms

(3) [the number of prime numbers]
    Thread-1 Ranges: {1~10}..... Prime Count: 1796
    Thread-2 Ranges: {11~20}..... Prime Count: 1795
    Thread-3 Ranges: {21~30}..... Prime Count: 1810
    Thread-4 Ranges: {31~40}..... Prime Count: 1808
    Thread-5 Ranges: {41~49}..... Prime Count: 1796
    Thread-6 Ranges: ..... Prime Count: 1789
    Thread-7 Ranges: ..... Prime Count: 1777
    Thread-8 Ranges: ..... Prime Count: 1796
    Thread-9 Ranges: ..... Prime Count: 1785
    Thread-10 Ranges: ..... Prime Count: 1832

1 ... 199999 prime# counter = 17984

-----------------------------
```

[Entire Threads : 12]

```
Entire Threads : 12

(1) [execution time of each thread]
    Thread-1 Execution Time : 2ms
    Thread-2 Execution Time : 5ms
    Thread-3 Execution Time : 1ms
    Thread-4 Execution Time : 2ms
    Thread-5 Execution Time : 4ms
    Thread-6 Execution Time : 2ms
    Thread-7 Execution Time : 3ms
    Thread-8 Execution Time : 3ms
    Thread-9 Execution Time : 2ms
    Thread-10 Execution Time : 1ms
    Thread-11 Execution Time : 3ms
    Thread-12 Execution Time : 1ms

(2) [program execution time]
    Program Execution Time : 6ms

(3) [the number of prime numbers]
    Thread-1 Ranges: {1~10}..... Prime Count: 1122
    Thread-2 Ranges: {11~20}..... Prime Count: 2239
    Thread-3 Ranges: {21~30}..... Prime Count: 1104
    Thread-4 Ranges: {31~40}..... Prime Count: 1129
    Thread-5 Ranges: {41~49}..... Prime Count: 2235
    Thread-6 Ranges: ..... Prime Count: 1157
    Thread-7 Ranges: ..... Prime Count: 1109
    Thread-8 Ranges: ..... Prime Count: 2268
    Thread-9 Ranges: ..... Prime Count: 1127
    Thread-10 Ranges: ..... Prime Count: 1123
    Thread-11 Ranges: ..... Prime Count: 2267
    Thread-12 Ranges: ..... Prime Count: 1104

1 ... 199999 prime# counter = 17984

-----------------------------
```

[Entire Threads : 14]

```
Entire Threads : 14

(1) [execution time of each thread]
    Thread-1 Execution Time : 3ms
    Thread-2 Execution Time : 1ms
    Thread-3 Execution Time : 2ms
    Thread-4 Execution Time : 2ms
    Thread-5 Execution Time : 3ms
    Thread-6 Execution Time : 3ms
    Thread-7 Execution Time : 3ms
    Thread-8 Execution Time : 2ms
    Thread-9 Execution Time : 2ms
    Thread-10 Execution Time : 2ms
    Thread-11 Execution Time : 3ms
    Thread-12 Execution Time : 1ms
    Thread-13 Execution Time : 2ms
    Thread-14 Execution Time : 2ms

(2) [program execution time]
    Program Execution Time : 6ms

(3) [the number of prime numbers]
    Thread-1 Ranges: {1~10}..... Prime Count: 1110
    Thread-2 Ranges: {11~20}..... Prime Count: 1516
    Thread-3 Ranges: {21~30}..... Prime Count: 1119
    Thread-4 Ranges: {31~40}..... Prime Count: 1494
    Thread-5 Ranges: {41~49}..... Prime Count: 1134
    Thread-6 Ranges: ..... Prime Count: 1506
    Thread-7 Ranges: ..... Prime Count: 1120
    Thread-8 Ranges: ..... Prime Count: 1127
    Thread-9 Ranges: ..... Prime Count: 1501
    Thread-10 Ranges: ..... Prime Count: 1128
    Thread-11 Ranges: ..... Prime Count: 1495
    Thread-12 Ranges: ..... Prime Count: 1109
    Thread-13 Ranges: ..... Prime Count: 1485
    Thread-14 Ranges: ..... Prime Count: 1140

1 ... 199999 prime# counter = 17984

-----------------------------
```

[Entire Threads : 16]

```
Entire Threads : 16

(1) [execution time of each thread]
    Thread-1 Execution Time : 1ms
    Thread-2 Execution Time : 2ms
    Thread-3 Execution Time : 2ms
    Thread-4 Execution Time : 2ms
    Thread-5 Execution Time : 2ms
    Thread-6 Execution Time : 3ms
    Thread-7 Execution Time : 2ms
    Thread-8 Execution Time : 3ms
    Thread-9 Execution Time : 2ms
    Thread-10 Execution Time : 3ms
    Thread-11 Execution Time : 2ms
    Thread-12 Execution Time : 2ms
    Thread-13 Execution Time : 1ms
    Thread-14 Execution Time : 1ms
    Thread-15 Execution Time : 2ms
    Thread-16 Execution Time : 2ms

(2) [program execution time]
    Program Execution Time : 7ms

(3) [the number of prime numbers]
    Thread-1 Ranges: {1~10}..... Prime Count: 1114
    Thread-2 Ranges: {11~20}..... Prime Count: 1146
    Thread-3 Ranges: {21~30}..... Prime Count: 1134
    Thread-4 Ranges: {31~40}..... Prime Count: 1122
    Thread-5 Ranges: {41~49}..... Prime Count: 1121
    Thread-6 Ranges: ..... Prime Count: 1141
    Thread-7 Ranges: ..... Prime Count: 1124
    Thread-8 Ranges: ..... Prime Count: 1127
    Thread-9 Ranges: ..... Prime Count: 1130
    Thread-10 Ranges: ..... Prime Count: 1105
    Thread-11 Ranges: ..... Prime Count: 1104
    Thread-12 Ranges: ..... Prime Count: 1124
    Thread-13 Ranges: ..... Prime Count: 1119
    Thread-14 Ranges: ..... Prime Count: 1127
    Thread-15 Ranges: ..... Prime Count: 1118
    Thread-16 Ranges: ..... Prime Count: 1128

1 ... 199999 prime# counter = 17984
```

[Entire Threads : 32]

```
Entire Threads : 32

(1) [execution time of each thread]
    Thread-1 Execution Time : 1ms
    Thread-2 Execution Time : 1ms
    Thread-3 Execution Time : 1ms
    Thread-4 Execution Time : 1ms
    Thread-5 Execution Time : 2ms
    Thread-6 Execution Time : 1ms
    Thread-7 Execution Time : 1ms
    Thread-8 Execution Time : 1ms
    Thread-9 Execution Time : 2ms
    Thread-10 Execution Time : 2ms
    Thread-11 Execution Time : 1ms
    Thread-12 Execution Time : 2ms
    Thread-13 Execution Time : 1ms
    Thread-14 Execution Time : 0ms
    Thread-15 Execution Time : 1ms
    Thread-16 Execution Time : 1ms
    Thread-17 Execution Time : 0ms
    Thread-18 Execution Time : 1ms
    Thread-19 Execution Time : 1ms
    Thread-20 Execution Time : 2ms
    Thread-21 Execution Time : 1ms
    Thread-22 Execution Time : 1ms
    Thread-23 Execution Time : 0ms
    Thread-24 Execution Time : 1ms
    Thread-25 Execution Time : 1ms
    Thread-26 Execution Time : 1ms
    Thread-27 Execution Time : 0ms
    Thread-28 Execution Time : 1ms
    Thread-29 Execution Time : 1ms
    Thread-30 Execution Time : 0ms
    Thread-31 Execution Time : 1ms
    Thread-32 Execution Time : 1ms

(2) [program execution time]
    Program Execution Time : 8ms

(3) [the number of prime numbers]
    Thread-1 Ranges: {1~10}..... Prime Count: 553
    Thread-2 Ranges: {11~20}..... Prime Count: 567
    Thread-3 Ranges: {21~30}..... Prime Count: 565
    Thread-4 Ranges: {31~40}..... Prime Count: 561
    Thread-5 Ranges: {41~49}..... Prime Count: 548
    Thread-6 Ranges: ..... Prime Count: 584
    Thread-7 Ranges: ..... Prime Count: 558
    Thread-8 Ranges: ..... Prime Count: 556
    Thread-9 Ranges: ..... Prime Count: 557
    Thread-10 Ranges: ..... Prime Count: 550
    Thread-11 Ranges: ..... Prime Count: 560
    Thread-12 Ranges: ..... Prime Count: 546
    Thread-13 Ranges: ..... Prime Count: 561
    Thread-14 Ranges: ..... Prime Count: 569
    Thread-15 Ranges: ..... Prime Count: 571
    Thread-16 Ranges: ..... Prime Count: 564
    Thread-17 Ranges: ..... Prime Count: 561
    Thread-18 Ranges: ..... Prime Count: 579
    Thread-19 Ranges: ..... Prime Count: 569
    Thread-20 Ranges: ..... Prime Count: 561
    Thread-21 Ranges: ..... Prime Count: 573
    Thread-22 Ranges: ..... Prime Count: 557
    Thread-23 Ranges: ..... Prime Count: 566
    Thread-24 Ranges: ..... Prime Count: 571
    Thread-25 Ranges: ..... Prime Count: 573
    Thread-26 Ranges: ..... Prime Count: 555
    Thread-27 Ranges: ..... Prime Count: 544
    Thread-28 Ranges: ..... Prime Count: 578
    Thread-29 Ranges: ..... Prime Count: 558
    Thread-30 Ranges: ..... Prime Count: 558
    Thread-31 Ranges: ..... Prime Count: 547
    Thread-32 Ranges: ..... Prime Count: 564

1 ... 199999 prime# counter = 17984
```

3. Dynamic load balancing

My dynamic load balancing method uses multithreading to find prime numbers between 1 and 200,000, and measures the execution time of each thread and the total program time. Through the given entire threads, the number of working threads is changed and the change results are checked. Depending on the number of threads, parallel processing is performed to find prime numbers, and each thread operates in separate ranges. This method can efficiently handle unbalanced workloads that occur during runtime. Moreover, due to the dynamically allocated tasks, the overall task processing speed of the system is increased

3-1) How to compile & execute

To create the source code, I first created a project in Eclipse, then created a project file called "Thread test." Next, I created a package called "Problem1_dynamic" I then created a class named "pc_dynamic" I wrote the code in this class, and to compile and execute it, I clicked on "Run." As a result, the program outputs the number of primes, the time taken by each thread, and the total program execution time, as shown above.

3-2) Result screen

[Entire Threads : 1]

```
Entire Threads : 1

(1) [execution time of each thread]
    Thread-1 Execution Time : 42ms

(2) [program execution time]
    Program Execution Time : 44ms

(3) [the number of prime numbers]
    Thread-1 Prime Count : 17984

1 ... 199999 prime# counter = 17984
```

[Entire Threads : 2]

```
Entire Threads : 2

(1) [execution time of each thread]
    Thread-1 Execution Time : 16ms
    Thread-2 Execution Time : 15ms

(2) [program execution time]
    Program Execution Time : 17ms

(3) [the number of prime numbers]
    Thread-1 Prime Count : 9144
    Thread-2 Prime Count : 8840

1 ... 199999 prime# counter = 17984
```

[Entire Threads : 4]

```
Entire Threads : 4

(1) [execution time of each thread]
    Thread-1 Execution Time : 8ms
    Thread-2 Execution Time : 7ms
    Thread-3 Execution Time : 7ms
    Thread-4 Execution Time : 7ms

(2) [program execution time]
    Program Execution Time : 8ms

(3) [the number of prime numbers]
    Thread-1 Prime Count : 5659
    Thread-2 Prime Count : 4746
    Thread-3 Prime Count : 3629
    Thread-4 Prime Count : 3950

1 ... 199999 prime# counter = 17984
```

[Entire Threads : 6]

```
Entire Threads : 6

(1) [execution time of each thread]
    Thread-1 Execution Time : 11ms
    Thread-2 Execution Time : 10ms
    Thread-3 Execution Time : 10ms
    Thread-4 Execution Time : 10ms
    Thread-5 Execution Time : 10ms
    Thread-6 Execution Time : 10ms

(2) [program execution time]
    Program Execution Time : 11ms

(3) [the number of prime numbers]
    Thread-1 Prime Count : 3123
    Thread-2 Prime Count : 3347
    Thread-3 Prime Count : 3239
    Thread-4 Prime Count : 2899
    Thread-5 Prime Count : 2321
    Thread-6 Prime Count : 3055

1 ... 199999 prime# counter = 17984
```

[Entire Threads : 8]

```
Entire Threads : 8

(1) [execution time of each thread]
    Thread-1 Execution Time : 10ms
    Thread-2 Execution Time : 9ms
    Thread-3 Execution Time : 8ms
    Thread-4 Execution Time : 9ms
    Thread-5 Execution Time : 9ms
    Thread-6 Execution Time : 8ms
    Thread-7 Execution Time : 7ms
    Thread-8 Execution Time : 8ms

(2) [program execution time]
    Program Execution Time : 10ms

(3) [the number of prime numbers]
    Thread-1 Prime Count : 3046
    Thread-2 Prime Count : 2310
    Thread-3 Prime Count : 2215
    Thread-4 Prime Count : 2373
    Thread-5 Prime Count : 1637
    Thread-6 Prime Count : 2128
    Thread-7 Prime Count : 1600
    Thread-8 Prime Count : 2675

1 ... 199999 prime# counter = 17984
```

[Entire Threads : 10]

```
Entire Threads : 10

(1) [execution time of each thread]
    Thread-1 Execution Time : 12ms
    Thread-2 Execution Time : 12ms
    Thread-3 Execution Time : 10ms
    Thread-4 Execution Time : 11ms
    Thread-5 Execution Time : 10ms
    Thread-6 Execution Time : 11ms
    Thread-7 Execution Time : 11ms
    Thread-8 Execution Time : 10ms
    Thread-9 Execution Time : 10ms
    Thread-10 Execution Time : 9ms

(2) [program execution time]
    Program Execution Time : 12ms

(3) [the number of prime numbers]
    Thread-1 Prime Count : 3125
    Thread-2 Prime Count : 2954
    Thread-3 Prime Count : 1524
    Thread-4 Prime Count : 2139
    Thread-5 Prime Count : 1819
    Thread-6 Prime Count : 750
    Thread-7 Prime Count : 1037
    Thread-8 Prime Count : 1077
    Thread-9 Prime Count : 2814
    Thread-10 Prime Count : 745

1 ... 199999 prime# counter = 17984
```

[Entire Threads : 12]

```
Entire Threads : 12

(1) [execution time of each thread]
    Thread-1 Execution Time : 10ms
    Thread-2 Execution Time : 10ms
    Thread-3 Execution Time : 9ms
    Thread-4 Execution Time : 9ms
    Thread-5 Execution Time : 9ms
    Thread-6 Execution Time : 8ms
    Thread-7 Execution Time : 8ms
    Thread-8 Execution Time : 8ms
    Thread-9 Execution Time : 8ms
    Thread-10 Execution Time : 7ms
    Thread-11 Execution Time : 7ms
    Thread-12 Execution Time : 7ms

(2) [program execution time]
    Program Execution Time : 11ms

(3) [the number of prime numbers]
    Thread-1 Prime Count : 2287
    Thread-2 Prime Count : 3153
    Thread-3 Prime Count : 1959
    Thread-4 Prime Count : 1302
    Thread-5 Prime Count : 1404
    Thread-6 Prime Count : 924
    Thread-7 Prime Count : 769
    Thread-8 Prime Count : 1239
    Thread-9 Prime Count : 760
    Thread-10 Prime Count : 1591
    Thread-11 Prime Count : 965
    Thread-12 Prime Count : 1631

1 ... 199999 prime# counter = 17984
```

[Entire Threads : 14]

```
Entire Threads : 14

(1) [execution time of each thread]
    Thread-1 Execution Time : 10ms
    Thread-2 Execution Time : 9ms
    Thread-3 Execution Time : 9ms
    Thread-4 Execution Time : 9ms
    Thread-5 Execution Time : 9ms
    Thread-6 Execution Time : 9ms
    Thread-7 Execution Time : 9ms
    Thread-8 Execution Time : 9ms
    Thread-9 Execution Time : 8ms
    Thread-10 Execution Time : 8ms
    Thread-11 Execution Time : 8ms
    Thread-12 Execution Time : 8ms
    Thread-13 Execution Time : 8ms
    Thread-14 Execution Time : 8ms

(2) [program execution time]
    Program Execution Time : 10ms

(3) [the number of prime numbers]
    Thread-1 Prime Count : 2772
    Thread-2 Prime Count : 1905
    Thread-3 Prime Count : 1469
    Thread-4 Prime Count : 1511
    Thread-5 Prime Count : 1394
    Thread-6 Prime Count : 1345
    Thread-7 Prime Count : 913
    Thread-8 Prime Count : 942
    Thread-9 Prime Count : 1252
    Thread-10 Prime Count : 880
    Thread-11 Prime Count : 757
    Thread-12 Prime Count : 1192
    Thread-13 Prime Count : 741
    Thread-14 Prime Count : 911

1 ... 199999 prime# counter = 17984
```

[Entire Threads : 32]

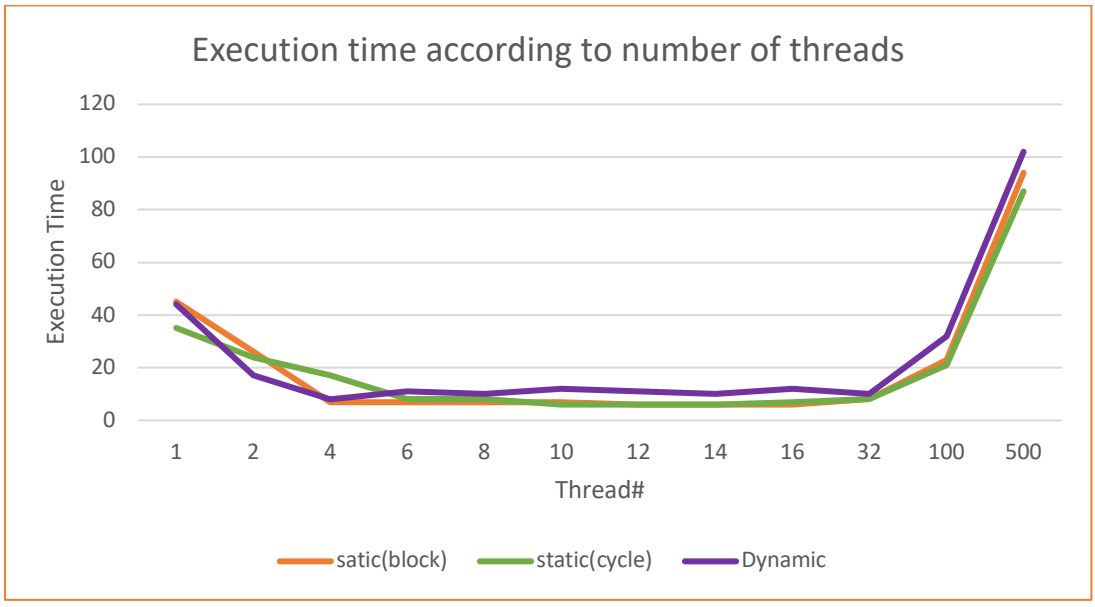
```
Entire Threads : 32

(1) [execution time of each thread]
    Thread-1 Execution Time : 9ms
    Thread-2 Execution Time : 10ms
    Thread-3 Execution Time : 9ms
    Thread-4 Execution Time : 10ms
    Thread-5 Execution Time : 10ms
    Thread-6 Execution Time : 9ms
    Thread-7 Execution Time : 9ms
    Thread-8 Execution Time : 9ms
    Thread-9 Execution Time : 8ms
    Thread-10 Execution Time : 8ms
    Thread-11 Execution Time : 8ms
    Thread-12 Execution Time : 8ms
    Thread-13 Execution Time : 8ms
    Thread-14 Execution Time : 7ms
    Thread-15 Execution Time : 7ms
    Thread-16 Execution Time : 6ms
    Thread-17 Execution Time : 7ms
    Thread-18 Execution Time : 6ms
    Thread-19 Execution Time : 7ms
    Thread-20 Execution Time : 6ms
    Thread-21 Execution Time : 6ms
    Thread-22 Execution Time : 6ms
    Thread-23 Execution Time : 6ms
    Thread-24 Execution Time : 6ms
    Thread-25 Execution Time : 6ms
    Thread-26 Execution Time : 6ms
    Thread-27 Execution Time : 6ms
    Thread-28 Execution Time : 5ms
    Thread-29 Execution Time : 4ms
    Thread-30 Execution Time : 5ms
    Thread-31 Execution Time : 5ms
    Thread-32 Execution Time : 4ms
```

```
(2) [program execution time]
    Program Execution Time : 10ms

(3) [the number of prime numbers]
    Thread-1 Prime Count : 2141
    Thread-2 Prime Count : 1361
    Thread-3 Prime Count : 863
    Thread-4 Prime Count : 940
    Thread-5 Prime Count : 899
    Thread-6 Prime Count : 1121
    Thread-7 Prime Count : 1014
    Thread-8 Prime Count : 565
    Thread-9 Prime Count : 529
    Thread-10 Prime Count : 180
    Thread-11 Prime Count : 293
    Thread-12 Prime Count : 627
    Thread-13 Prime Count : 499
    Thread-14 Prime Count : 767
    Thread-15 Prime Count : 611
    Thread-16 Prime Count : 529
    Thread-17 Prime Count : 232
    Thread-18 Prime Count : 434
    Thread-19 Prime Count : 375
    Thread-20 Prime Count : 311
    Thread-21 Prime Count : 230
    Thread-22 Prime Count : 115
    Thread-23 Prime Count : 743
    Thread-24 Prime Count : 212
    Thread-25 Prime Count : 271
    Thread-26 Prime Count : 113
    Thread-27 Prime Count : 370
    Thread-28 Prime Count : 590
    Thread-29 Prime Count : 361
    Thread-30 Prime Count : 204
    Thread-31 Prime Count : 223
    Thread-32 Prime Count : 261

1 ... 199999 prime# counter = 17984

-----------------------------
```

4. table and graph

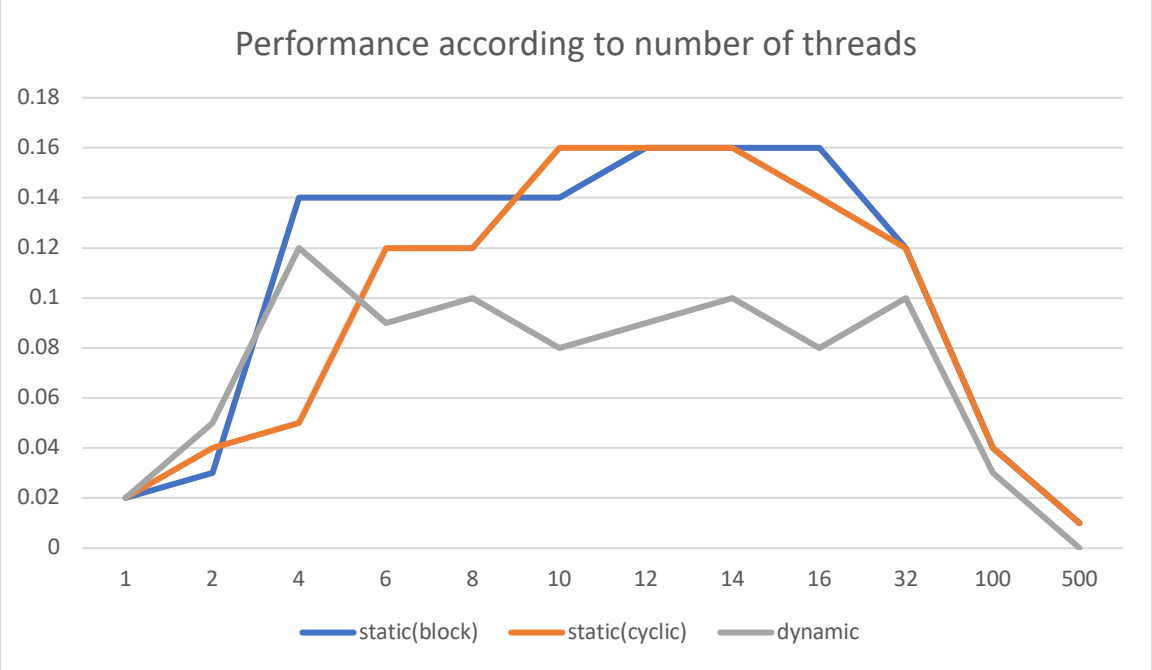
Exec Time	1	2	4	6	8	10
Static(block)	45ms	26ms	7ms	7ms	7ms	7ms
Static(cyclic) [task size : 10 numbers]	35ms	24ms	17ms	8ms	8ms	6ms
Dynamic [task size : 10 numbers]	44ms	17ms	8ms	11mss	10ms	12ms

Exec Time	12	14	16	32	100	500
Static(block)	6ms	7ms	6ms	8ms	23ms	94ms
Static(cyclic) [task size : 10 numbers]	6ms	6ms	7ms	8ms	21ms	87ms
Dynamic [task size : 10 numbers]	11ms	10ms	12mss	10ms	32ms	102ms



Performance	1	2	4	6	8	10
Static(block)	0.02ms	0.03ms	0.14ms	0.14ms	0.14ms	0.14ms
Static(cyclic) [task size : 10 numbers]	0.02ms	0.04ms	0.05ms	0.12ms	0.12ms	0.16ms
Dynamic [task size : 10 numbers]	0.02ms	0.05ms	0.12ms	0.09mss	0.1ms	0.08ms

Performance	12	14	16	32	100	500
Static(block)	0.16ms	0.16ms	0.16ms	0.12ms	0.04ms	0.01ms
Static(cyclic) [task size : 10 numbers]	0.16ms	0.16ms	0.14ms	0.12ms	0.04ms	0.01ms
Dynamic [task size : 10 numbers]	0.09ms	0.1ms	0.03ms	0.1ms	0.03ms	0.009ms



5. explain/analysis on the result and why such result are obtained

Looking at the speed results of static load balancing (block decomposition), we can see that the execution time is noticeably reduced when the number of threads is increased from 1 to 2. The same is true when the number of threads is increased from 2 to 4. However, using 4, 6, 8, 10, 12, 14, 16, or 32 threads results in similar speeds. Similarly, static load balancing (cycle decomposition) and dynamic load balancing also show similar results. We can see that there is no significant improvement in the number of threads beyond a certain level. Thinking about the factors that may have influenced these results, we can consider Amdahl's Law. Performance improvement due to parallelization is limited by the sequential processing portion. Therefore, increasing the number of threads does not necessarily lead to better performance. There is little difference and hardly any change.

I also believe that the number of CPU cores affects the results. If more threads are created than the number of cores, performance improvements due to thread switching do not appear significant. Therefore, if the number of CPU cores is fewer than the number of threads, parallel processing performance can be limited.

To verify the changes in performance and speed, I tested with 100 and 500 threads. Despite the increase in the number of threads, performance and execution speed did not significantly decrease. From this, we can see that increasing the number of threads does not significantly improve execution speed and performance. I think this is due to the overhead caused by many threads. As the number of threads increases, the overhead required for thread management also increases, ultimately leading to performance degradation. Therefore, an optimal performance should be achieved using an appropriate number of threads.

JAVA source code of pc_static_bock

```java
package Problem1_block;

public class pc_static_block {


    private static int NUM_END = 200000;
    private static int[] NUM_THREADS = {1, 2, 4, 6, 8,
10, 12, 14, 16, 32,100,500};

    public static void main(String[] args) {

        for (int num_Threads : NUM_THREADS) {
          int counter = 0;
          int[] eachPrimeCnt = new int[num_Threads];
            long startTime = System.currentTimeMillis();


            Thread[] threads = new Thread[num_Threads];

            PrimeCounter[] primeCnt = new
PrimeCounter[num_Threads];


            int blockSize = NUM_END / num_Threads;
            for (int i = 0; i < num_Threads; i++) {
                int start = i * blockSize;
                int end = (i == num_Threads - 1) ?
NUM_END : (i + 1) * blockSize;
                primeCnt[i] = new PrimeCounter(start,
end);
                threads[i] = new Thread(primeCnt[i]);
                threads[i].start();
            }
```

```
            for (int i = 0; i < num_Threads; i++) {
                try {
                    threads[i].join();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                eachPrimeCnt[i] =
primeCnt[i].returnPrimeCnt();
                counter += primeCnt[i].returnPrimeCnt();
            }

            long endTime = System.currentTimeMillis();
            long timeDiff = endTime - startTime;
            System.out.println("Entire Threads : " +
num_Threads);
            System.out.println();


            System.out.println("(1) [execution time of
each thread]");

            for(int i = 0;i<num_Threads;i++) {
                System.out.println("    Thread-"+ (i + 1)
+ " Execution Time : " + primeCnt[i].returnThreadTime() +
"ms");
            }
            System.out.println();


            System.out.println("(2) [program execution
time]");
            System.out.println("    Program Execution
Time : " + timeDiff + "ms");
            System.out.println();
```

```
            System.out.println("(3) [the number of prime
numbers]");
            for(int i = 0; i < num_Threads; i++) {
                System.out.println("    Thread-" + (i +
1) + " Range: " + primeCnt[i].returnStart() + " to " +
(primeCnt[i].returnEnd() - 1) + ", Prime Count: " +
eachPrimeCnt[i]);
            }

            System.out.println();

            System.out.println("1..." + (NUM_END-1) + "
prime# counter = " + counter);

            System.out.println();
            System.out.println("-------------------------
-----");
            System.out.println();
        }
    }


    private static class PrimeCounter implements Runnable
{
        private int start;
        private int end;
        private int count;
        private long threadTime;


        public PrimeCounter(int start, int end) {
            this.start = start;
            this.end = end;
            this.count = 0;
        }
```

```
        public int returnPrimeCnt() {
            return count;
        }

        public long returnThreadTime() {
            return threadTime;
        }

        public int returnStart() {
            return start;
        }

        public int returnEnd() {
            return end;
        }


        public void run() {
            long startThreadTime =
System.currentTimeMillis();
            for (int i = start; i < end; i++) {
                if (isPrime(i)) {
                      count++;
                }
            }
            long endThreadTime =
System.currentTimeMillis();
            threadTime = endThreadTime - startThreadTime;
        }
        private static boolean isPrime(int x) {
            if (x <= 1) {
                return false;
            }
```

```
                if (x % i == 0) {
                    return false;
                }
            }
            return true;
        }
    }
}
```

JAVA source code of pc_static_cyclic

```java
package Problem1_cyclic;

public class pc_stack_cyclic {


    private static final int NUM_END = 200000;
    private static final int[] NUM_THREADS = {1, 2, 4, 6,
8, 10, 12, 14, 16, 32};

    public static void main(String[] args) {

        for (int num_Threads : NUM_THREADS) {
          int[] eachPrimeCnt = new int[num_Threads];
          int counter = 0;
            long startTime = System.currentTimeMillis();


            Thread[] threads = new Thread[num_Threads];

            PrimeCounter[] primeCnt = new
PrimeCounter[num_Threads];
            for (int i = 0; i < num_Threads; i++) {
                primeCnt[i] = new PrimeCounter(i,
num_Threads);
                threads[i] = new Thread(primeCnt[i]);
                threads[i].start();
            }

            for (int i = 0; i < num_Threads; i++) {
                try {
                    threads[i].join();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
```

```
                eachPrimeCnt[i] =
primeCnt[i].returnPrimeCnt();
                counter += eachPrimeCnt[i];
            }

            long endTime = System.currentTimeMillis();
            long timeDiff = endTime - startTime;

            System.out.println("Entire Threads : " +
num_Threads);
            System.out.println();


            System.out.println("(1) [execution time of
each thread]");

            for(int i = 0;i<num_Threads;i++) {
                System.out.println("    Thread-"+ (i + 1)
+ " Execution Time : " + primeCnt[i].returnThreadtime() +
"ms");
            }
            System.out.println();


            System.out.println("(2) [program execution
time]");
            System.out.println("    Program Execution
Time : " + timeDiff + "ms");
            System.out.println();


            System.out.println("(3) [the number of prime
numbers]");
            for(int i = 0; i < num_Threads; i++) {
```

```
                System.out.print("    Thread-" + (i + 1)
+ " Ranges: ");
                int current = primeCnt[i].returnStart() *
10 + 1;
                while (current < 500) {
                    int endPoint = Math.min(current + 10,
500);
                    System.out.print("{" + current + "~"
+ (endPoint - 1) + "}");
                    if (current +
primeCnt[i].returnStep() * 10 < 500) {
                        System.out.print(", ");
                    }
                    current += primeCnt[i].returnStep() *
10;
                }
                System.out.print("......");
                System.out.println(" Prime Count: " +
eachPrimeCnt[i]);
            }

            System.out.println();
            System.out.println("1..." + (NUM_END-1) + "
prime# counter = " + counter);

            System.out.println();
            System.out.println("----------------------
-----");
            System.out.println();
        }
    }


    private static class PrimeCounter implements Runnable
{
```

```
        private int start;
        private int   step;
        private int count;
        private long eachThreadtime;

        public PrimeCounter(int start, int step) {
            this.start = start;
            this.step = step;
            this.count = 0;
        }

        public int returnPrimeCnt() {
            return count;
        }

        public long returnThreadtime() {
          return eachThreadtime;
        }

        public int returnStart() {
            return start;
        }

        public int returnStep() {
            return step;
        }


        public void run() {
            long startThreadtime =
System.currentTimeMillis();
            int current = start * 10 + 1;
            while (current < NUM_END) {
                int endPoint = Math.min(current + 10,
NUM_END);
```

```
                for (int i = current; i < endPoint; i++)
{
                    if (isPrime(i)) {
                        count++;
                    }
                }
                current += step * 10;
            }
            long endThreadtime =
System.currentTimeMillis();
            eachThreadtime = endThreadtime -
startThreadtime;
        }



        private static boolean isPrime(int x) {
            if (x <= 1) {
                return false;
            }


            for (int i = 2; i *i<= x; i++) {
                if (x % i == 0) {
                    return false;
                }
            }
            return true;
        }
    }
}
```

JAVA source code of pc_dynamic

```java
package Problem1_dynamic;

public class pc_dynamic {
     private static final int NUM_END = 200000;
        private static final int[] NUM_THREADS = {1, 2,
4, 6, 8, 10, 12, 14, 16, 32};

        public static void main(String[] args) {
            for (int num_Threads : NUM_THREADS) {
                int[] eachPrimeCnt = new
int[num_Threads];
                int counter = 0;
                long startTime =
System.currentTimeMillis();

                Thread[] threads = new
Thread[num_Threads];
                PrimeCounter[] primeCnt = new
PrimeCounter[num_Threads];

                ThreadManager threadManager = new
ThreadManager();

                for (int i = 0; i < num_Threads; i++) {
                    primeCnt[i] = new
PrimeCounter(threadManager);
                    threads[i] = new Thread(primeCnt[i]);
                    threads[i].start();
                }

                for (int i = 0; i < num_Threads; i++) {
                    try {
                        threads[i].join();
                    } catch (InterruptedException e) {
                        e.printStackTrace();
```

```
                    }
                    eachPrimeCnt[i] =
primeCnt[i].returnPrimeCnt();
                    counter += eachPrimeCnt[i];
                }

                long endTime =
System.currentTimeMillis();
                long timeDiff = endTime - startTime;

                System.out.println("Entire Threads : " +
num_Threads);
                System.out.println();


                System.out.println("(1) [execution time
of each thread]");

                for(int i = 0;i<num_Threads;i++) {
                    System.out.println("    Thread-"+ (i
+ 1) + " Execution Time : " +
primeCnt[i].returnThreadtime() + "ms");
                }
                System.out.println();


                System.out.println("(2) [program
execution time]");
                System.out.println("    Program Execution
Time : " + timeDiff + "ms");
                System.out.println();

                System.out.println("(3) [the number of
prime numbers]");
                for(int i = 0; i<num_Threads; i++) {
```

```
                    System.out.println("    Thread-" +
(i + 1) + " Prime Count : " + eachPrimeCnt[i]);
                }

                System.out.println();
                System.out.println("1..." + (NUM_END-1) +
" prime# counter = " + counter);

                System.out.println();
                System.out.println("----------------------
---------");
                System.out.println();
            }
        }

        private static class ThreadManager {
            private int currentTask;
            private int taskSize = 10;


            public ThreadManager() {
                currentTask = 1;
            }


            public synchronized int returnNextTask() {
                int task = currentTask;
                currentTask += taskSize;
                return task;
            }
        }

        private static class PrimeCounter implements
Runnable {
            private ThreadManager threadManager;
```

```
            private int count;
            private long eachThreadtime;

            public PrimeCounter(ThreadManager
taskManager) {
                this.threadManager = taskManager;
                this.count = 0;
            }

            public int returnPrimeCnt() {
                return count;
            }

            public long returnThreadtime() {
                return eachThreadtime;
            }

            public void run() {
                long startThreadtime =
System.currentTimeMillis();

                while (true) {
                    int current =
threadManager.returnNextTask();
                    if (current >= NUM_END) {
                        break;
                    }
                    int endPoint = Math.min(current + 10,
NUM_END);
                    for (int i = current; i < endPoint;
i++) {
                        if (isPrime(i)) {
                                count++;
                        }
                    }
```

```
                }

                long endThreadtime =
System.currentTimeMillis();
                eachThreadtime = endThreadtime -
startThreadtime;
            }

            private static boolean isPrime(int x) {
                if (x <= 1) {
                    return false;
                }

                for (int i = 2; i * i <= x; i++) {
                    if (x % i == 0) {
                        return false;
                    }
                }
                return true;
            }
        }
    }
```