2022.1 Multicore Computing, Project #1

Problem 1

Document

소프트웨어학부

20176342 송민준

(i) Result

(a) Execution environment

CPU : AMD Ryzen 5 2600X Six-Core Processor (12 CPUs), ~3.6GHz

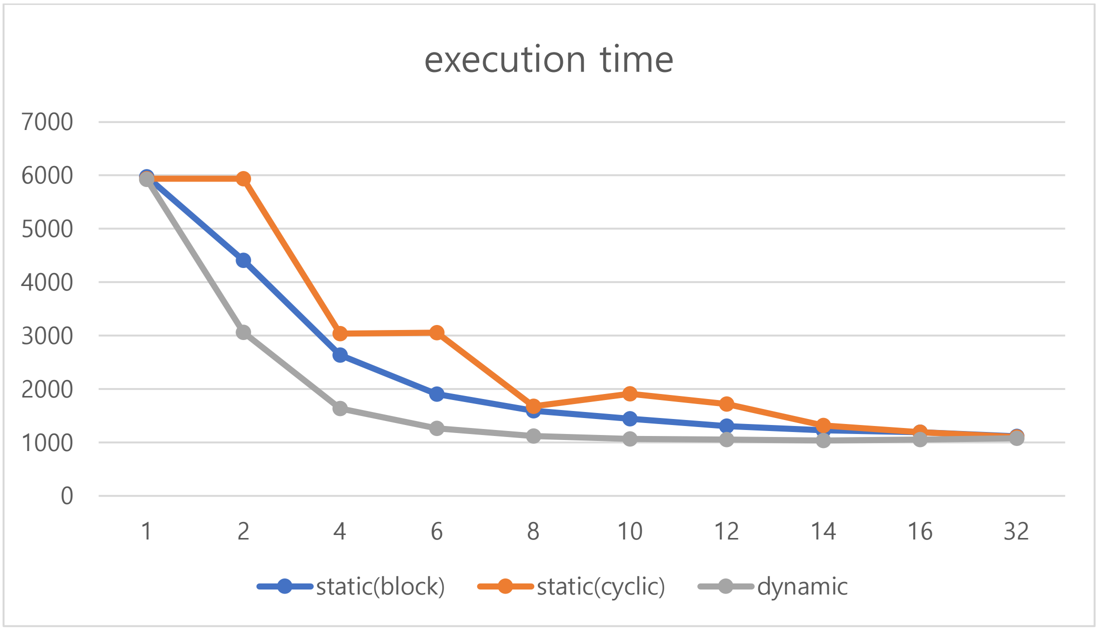
Memory : DDR4 16384MB RAM

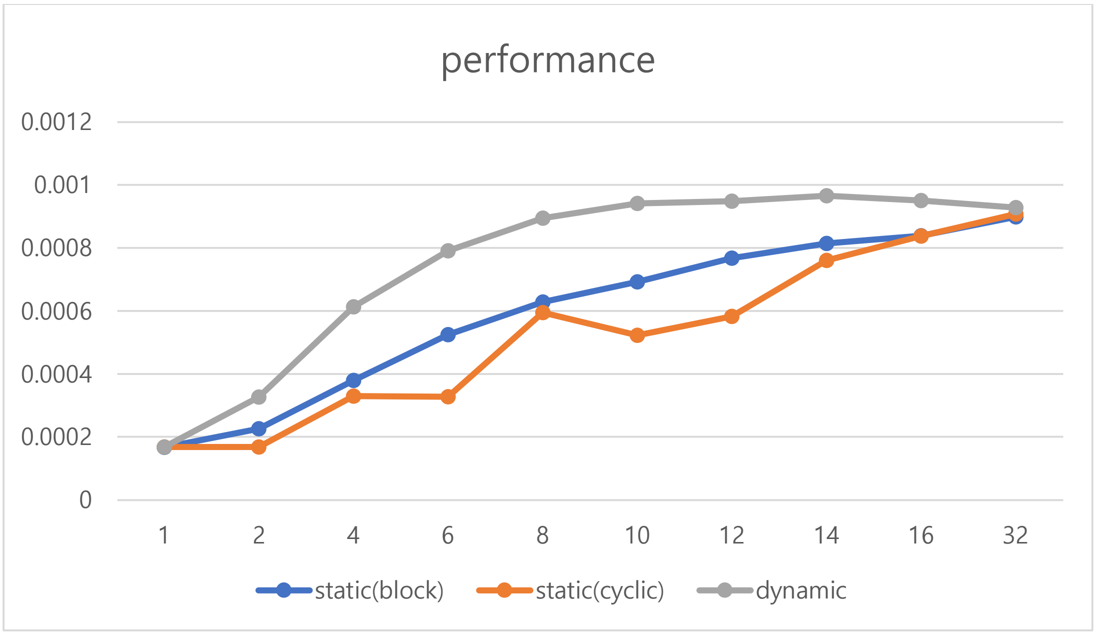
OS : Windows 10

(b) Tables and graphs

Exec time(ms)	1	2	4	6	8	10	12	14	16	32
Static (block)	5973	4410	2636	1904	1592	1444	1302	1227	1191	1112
Static (cyclic)	5935	5936	3036	3054	1678	1912	1715	1315	1192	1101
dynamic	5923	3061	1631	1264	1117	1062	1054	1035	1052	1077

Performance (1/exec time)	1	2	4	6	8	10	12	14	16	32
Static (block)	0.000167	0.000226	0.000379	0.000525	0.000628	0.000692	0.000768	0.000814	0.000839	0.000899
Static (cyclic)	0.000168	0.000168	0.000329	0.000327	0.000595	0.000523	0.000583	0.000760	0.000838	0.000908
dynamic	0.000168	0.000168	0.000168	0.000168	0.000168	0.000168	0.000168	0.000168	0.000168	0.000168


execution time
7000
6000
5000
4000
3000
2000
1000
0
1
2
4
6
8
10
12
14
16
32
static(block)
static(cyclic)
dynamic


performance
0.0012
0.001
0.0008
0.0006
0.0004
0.0002
0
1
2
4
6
8
10
12
14
16
32
static(block)
static(cyclic)
dynamic

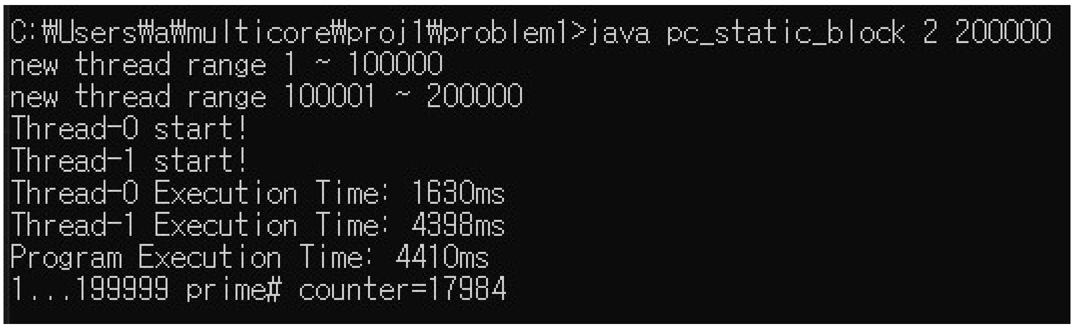
(c) Explanation of results

First, In static (block) load balancing method decreases execution time exponentially by increasing number of threads until almost 10 threads..

Because each thread doing a job(get a prime number in number range) same range.

For example, when there are threads 1 and 2, a range of 1 ~ 100000 and 100001 ~ 200000 is allocated, respectively.

However, in the function of finding prime numbers, the repetition statements are used to determine if they are divided by i from 1~n, so passing over large numbers by parameters takes longer to calculate.



```
C:₩Users₩a₩multicore₩proj1₩problem1>java pc_static_block 2 200000
new thread range 1 ~ 100000
new thread range 100001 ~ 200000
Thread-0 start!
Thread-1 start!
Thread-0 Execution Time: 1630ms
Thread-1 Execution Time: 4398ms
Program Execution Time: 4410ms
1...199999 prime# counter=17984
```

Above screen capture shows the result of static(block) method using 2 threads.

Thread-0 is assigned work number range 1~100000, and

Thread-1 is assigned work number range 100001~200000.

Consequently, Thread-0 execution time is about a third of the execution time of thread 1.

I can calculate number of iteration when work number range was given.

1~100000 is (1+100000)*(100000/2) = 5000050000

100001~200000 is (100001+200000)*(100000/2) = 15000050000

15000050000 / 5000050000 = almost 3

The execution time of Thread-1 is three times the execution time of Thread-0.However, from the time when there were more than 10 threads, the execution time was hardly reduced.

More threads cause a lot of overhead for "context switching." Therefore, it is important to find the appropriate number of threads, and in the static block method, the value seems to be about 10 to 12.

Second, In static (cyclic) load balancing method

In cyclic method, if there are 4 threads, each threads assigned their job(get a prime number from number)

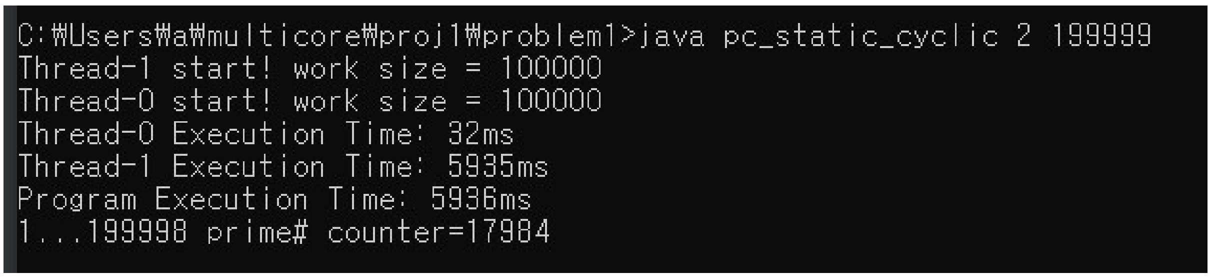
1,5,9,13,,,

2,6,10,14,,,

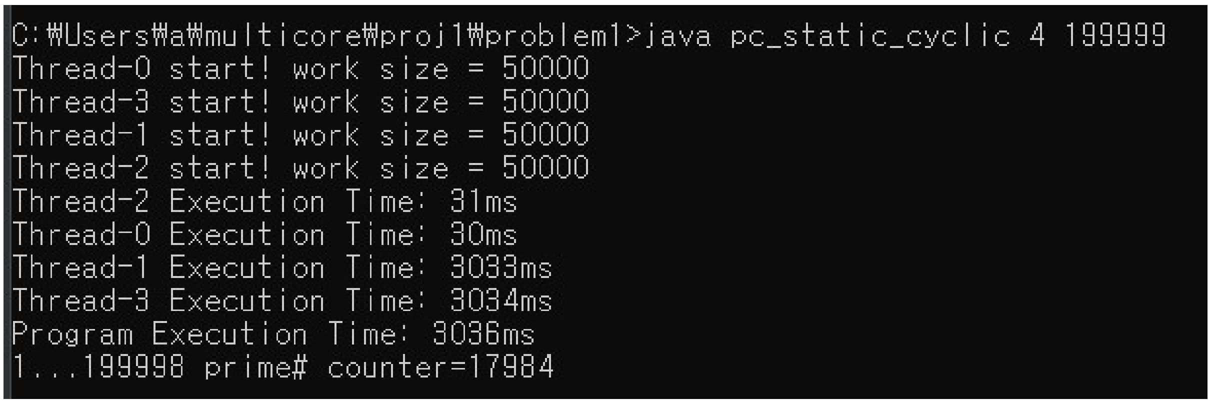
3,7,11,15,,,

4,8,12,16,,, respectively

Comparing when there are one thread and when there are two threads, the execution time is almost the same.



```
C:₩Users₩a₩multicore₩proj1₩problem1>java pc_static_cyclic 2 199999
Thread-1 start! work size = 100000
Thread-0 start! work size = 100000
Thread-0 Execution Time: 32ms
Thread-1 Execution Time: 5935ms
Program Execution Time: 5936ms
1...199998 prime# counter=17984
```



```
C:₩Users₩a₩multicore₩proj1₩problem1>java pc_static_cyclic 4 199999
Thread-0 start! work size = 50000
Thread-3 start! work size = 50000
Thread-1 start! work size = 50000
Thread-2 start! work size = 50000
Thread-2 Execution Time: 31ms
Thread-0 Execution Time: 30ms
Thread-1 Execution Time: 3033ms
Thread-3 Execution Time: 3034ms
Program Execution Time: 3036ms
1...199998 prime# counter=17984
```

The screenshot above is the result of executing when there are two threads and four threads.

When there are two threads, there is little execution time of thread 0. When there are four threads, there is little execution time for thread 0 and thread 2.

```
 private static boolean isPrime(int x){
   int i;
   if(x<=1) return false;
   for(i=2;i<x;i++){
     if(x%i == 0) return false;
   }
   return true;
 }
```

The reason lies in the internal logic of the isPrime function.

An even thread is always assigned an even number.

When an even number enters isPrime function as a parameter, the result of modular operation is always 0. And returns result immediately.(just 1 iteration only needed)

Therefore, in the case of 'static (cyclic) method', there is little difference between one thread and two threads.

Likewise, if there are too many threads, the context switching overhead increases and the execution time is no longer faster.

Finally, In dynamic load balancing method

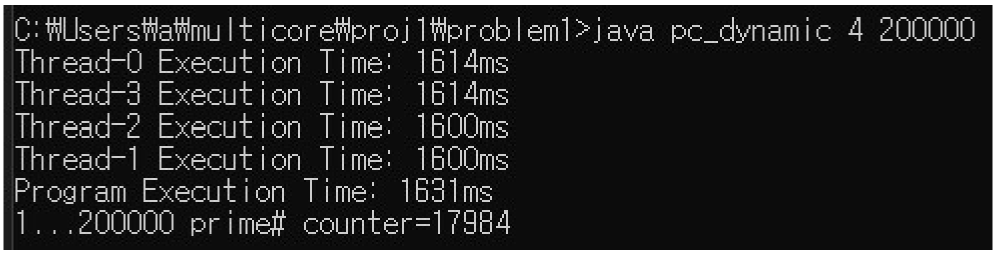
```java
public void work(){
  while(true){
    lock.lock();
    if(index >= end){
      lock.unlock();
      break;
    }
    int curr = p[index++];
    lock.unlock();
    if( isPrime(p[curr]) ){
      primeCount++;
    }


  }
}
```

Dynamic load balancing, unlike static load balancing, induces a natural state of competition among threads, so it seems that the balancing was performed most appropriately.

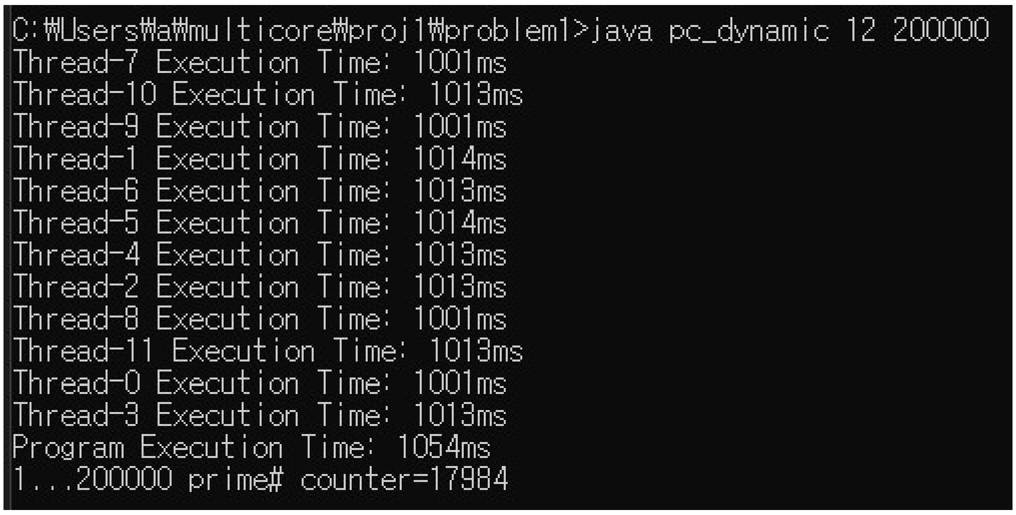
In dynamic load balancing, threads compete to lock and unlock by using lock and unlock of reentrant lock.

Unlike the static block method and the static cyclic method, there is no case in which an operation is advantageous only for a specific thread, and thus the result is the best compared to other load balancing methods.

Similarly, dynamic load balancing also increased context switching overhead when the number of threads was too high, and the execution time was no longer faster when the number of threads exceeded a certain number (approximately 10).



```
C:₩Users₩a₩multicore₩proj1₩problem1>java pc_dynamic 4 200000
Thread-0 Execution Time: 1614ms
Thread-3 Execution Time: 1614ms
Thread-2 Execution Time: 1600ms
Thread-1 Execution Time: 1600ms
Program Execution Time: 1631ms
1...200000 prime# counter=17984
```



```
C:\Users\a\multicore\proj1\problem1>java pc_dynamic 12 200000
Thread-7 Execution Time: 1001ms
Thread-10 Execution Time: 1013ms
Thread-9 Execution Time: 1001ms
Thread-1 Execution Time: 1014ms
Thread-6 Execution Time: 1013ms
Thread-5 Execution Time: 1014ms
Thread-4 Execution Time: 1013ms
Thread-2 Execution Time: 1013ms
Thread-8 Execution Time: 1001ms
Thread-11 Execution Time: 1013ms
Thread-0 Execution Time: 1001ms
Thread-3 Execution Time: 1013ms
Program Execution Time: 1054ms
1...200000 prime# counter=17984
```

As shown in the picture above, it can be seen that the execution time for each thread is not biased.

(d) entire JAVA source code and screen capture image of program execution and output

pc_static_block.java

```
import java.util.ArrayList;
import java.util.Arrays;

public class pc_static_block {
  private static int NUM_END = 200000;
  private static int NUM_THREADS = 1;
  public static void main (String[] args){
    if(args.length==2){
      NUM_THREADS = Integer.parseInt(args[0]);
      NUM_END = Integer.parseInt(args[1]);
    }

    int[] problem = new int[NUM_END];
    for(int i = 0; i<NUM_END;i++){
      problem[i] = i;
    }

    int counter = 0;
    long startTime = System.currentTimeMillis();

    ArrayList<BlockThread> thread_arr = new ArrayList<BlockThread>();

    for(int i = 0; i<NUM_THREADS;i++){
      int start = i*(NUM_END/NUM_THREADS)+1;
      int end = i == NUM_THREADS-1 ? NUM_END : (i+1)*(NUM_END/NUM_THREADS);
      System.out.println("new thread range "+start+ " ~ "+end);
      BlockThread a = new BlockThread(Arrays.copyOfRange(problem, start ,
end));
      thread_arr.add(a);
      a.start();

    }

    for(int i = 0;i<thread_arr.size();i++){
      try {
        thread_arr.get(i).join();
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
    }
```

```java
    for(int i = 0;i<thread_arr.size();i++){
      counter += thread_arr.get(i).getResult();
    }

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

    System.out.println("Program Execution Time: "+timeDiff+"ms");
    System.out.println("1..."+(NUM_END-1)+" prime# counter=" + counter);
  }

}

class BlockThread extends Thread {
  int[] problem;
  int primeCount = 0;
  long startTime = System.currentTimeMillis();


  BlockThread( int[] problem ){
    this.problem = problem;
  }

  public void run(){
    System.out.println(this.getName()+" start!");

    for(var i = 0; i<this.problem.length;i++){
      if(isPrime(this.problem[i])){
        primeCount++;
      }
    }

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

    System.out.println(this.getName()+" Execution Time: "+timeDiff+"ms");
  }

  public int getResult(){
    return this.primeCount;
  }

  private static boolean isPrime(int x){
    int i;
    if(x<=1) return false;
    for(i=2;i<x;i++){
      if(x%i == 0) return false;
    }
    return true;
  }
}
```

pc_static_cyclic.java

```java
import java.util.ArrayList;
import java.util.Arrays;

public class pc_static_cyclic {
  private static int NUM_END = 200000;
  private static int NUM_THREADS = 1;
  public static void main (String[] args){
    if(args.length==2){
      NUM_THREADS = Integer.parseInt(args[0]);
      NUM_END = Integer.parseInt(args[1]);
    }

    int[] problem = new int[NUM_END];
    for(int i = 0; i<NUM_END;i++){
      problem[i] = i;
    }

    int counter = 0;
    long startTime = System.currentTimeMillis();

    ArrayList<CyclicThread> thread_arr = new ArrayList<CyclicThread>();

    for(int i = 0; i<NUM_THREADS;i++){

      CyclicThread a = new CyclicThread();
      thread_arr.add(a);

    }


    int k = 0;

    while(k<=NUM_END){
      thread_arr.get(k%NUM_THREADS).addWork(k++);
    }

    for(int i = 0;i<thread_arr.size();i++){
      thread_arr.get(i).start();
    }
    for(int i = 0;i<thread_arr.size();i++){
      try {
        thread_arr.get(i).join();
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
    }

    for(int i = 0;i<thread_arr.size();i++){
      counter += thread_arr.get(i).getResult();
    }

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

    System.out.println("Program Execution Time: "+timeDiff+"ms");
    System.out.println("1..."+(NUM_END-1)+" prime# counter=" + counter);
  }

}

class CyclicThread extends Thread {
  ArrayList<Integer> problem;
  int primeCount = 0;
  long startTime = System.currentTimeMillis();


  CyclicThread( ){
    this.problem = new ArrayList<Integer>();
  }

  public void addWork(int num){
    this.problem.add(num);
  }

  public void run(){
    System.out.println(this.getName()+" start! work size =
"+this.problem.size());

    for(var i = 0; i<this.problem.size();i++){
      if(isPrime(this.problem.get(i))){
        primeCount++;
      }
    }

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

    System.out.println(this.getName()+" Execution Time: "+timeDiff+"ms");
  }
  public int getResult(){
    return this.primeCount;
  }

  private static boolean isPrime(int x){
    int i;
    if(x<=1) return false;
    for(i=2;i<x;i++){
      if(x%i == 0) return false;
    }
    return true;
  }
}
```

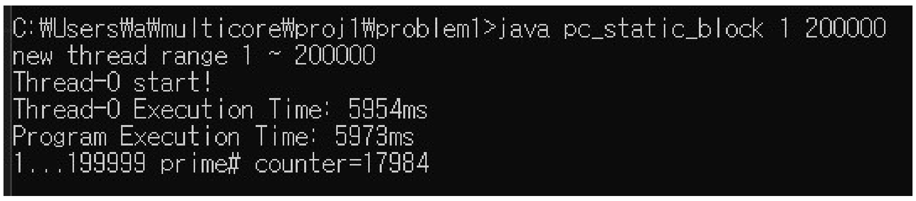
pc_dynamic.java

```java
import java.util.ArrayList;
import java.util.Arrays;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

public class pc_dynamic {
  private static int NUM_END = 200000;
  private static int NUM_THREADS = 1;
  public static void main (String[] args){
    if(args.length==2){
      NUM_THREADS = Integer.parseInt(args[0]);
      NUM_END = Integer.parseInt(args[1]);
    }

    int[] problem = new int[NUM_END];
    for(int i = 0; i<NUM_END;i++){
      problem[i] = i;
    }

    int counter = 0;
    long startTime = System.currentTimeMillis();

    ArrayList<DynamicThread> thread_arr = new ArrayList<DynamicThread>();

    final Lock lock = new ReentrantLock();

    for(int i = 0; i<NUM_THREADS;i++){

      DynamicThread a = new DynamicThread(lock);
      thread_arr.add(a);
    }


    DynamicThread.p = problem;
    DynamicThread.end = NUM_END;


    for(int i = 0;i<thread_arr.size();i++){
      thread_arr.get(i).start();
    }
    for(int i = 0;i<thread_arr.size();i++){
      try {
        thread_arr.get(i).join();
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
    }

    for(int i = 0;i<thread_arr.size();i++){
      counter += thread_arr.get(i).getResult();
    }

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

    System.out.println("Program Execution Time: "+timeDiff+"ms");
    System.out.println("1..."+(NUM_END)+" prime# counter=" + counter);
  }

}

class DynamicThread extends Thread {
  static ArrayList<Integer> problem = new ArrayList<Integer>();
  static int[] p;
  static int index = 0;
  static int end = 0;
  int primeCount = 0;
  long startTime = System.currentTimeMillis();
  private Lock lock;


  DynamicThread( Lock lock ){
    this.lock = lock;
  }

  public static void addWork(int num){
    problem.add(num);
```

```java
}

public void work(){
  while(true){
    lock.lock();
    if(index >= end){
      lock.unlock();
      break;
    }
    int curr = p[index++];
    lock.unlock();
    if( isPrime(p[curr]) ){
      primeCount++;
    }

  }
}


public int getWork(){
  lock.lock();

  if(problem.size()>0){
      int val;
      val = problem.get(0);
      problem.remove(0);
      lock.unlock();

      return val;
  }
  else {
    System.out.println(this.getName()+" empty");
  }
  lock.unlock();

  return -1;
}

public void run(){
  work();

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

  System.out.println(this.getName()+" Execution Time: "+timeDiff+"ms");
}

public int getResult(){
    return primeCount;
  }

  private boolean isPrime(int x){
    int i;
    if(x<=1) return false;
    for(i=2;i<x;i++){
      if(x%i == 0) return false;
    }
    return true;
  }
}
```

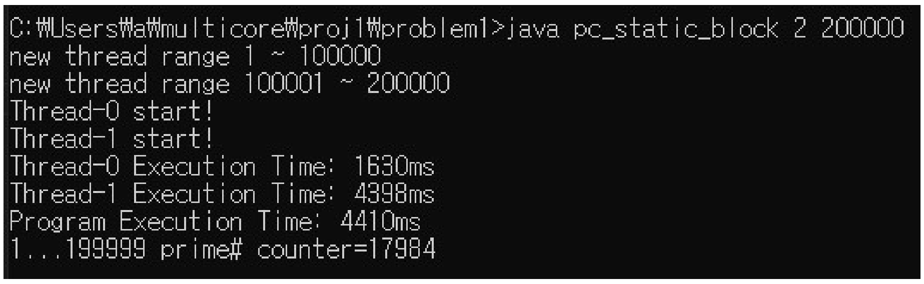
PC_static_block.java

pc_static_block thread #1



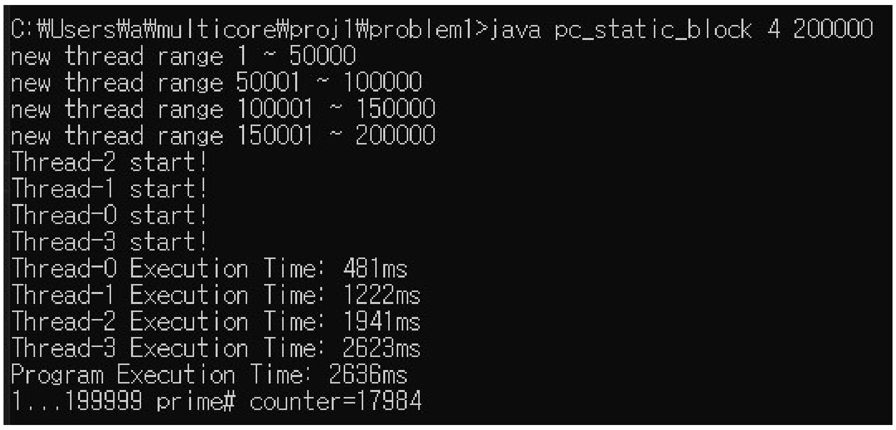
```
C:₩Users₩a₩multicore₩proj1₩problem1>java pc_static_block 1 200000
new thread range 1 ~ 200000
Thread-0 start!
Thread-0 Execution Time: 5954ms
Program Execution Time: 5973ms
1...199999 prime# counter=17984
```

pc_static_block thread #2

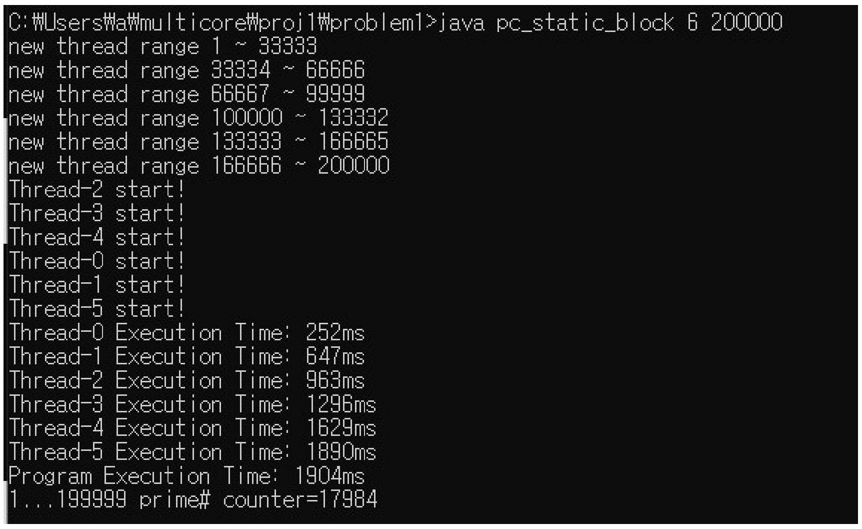


```
C:₩Users₩a₩multicore₩proj1₩problem1>java pc_static_block 2 200000
new thread range 1 ~ 100000
new thread range 100001 ~ 200000
Thread-0 start!
Thread-1 start!
Thread-0 Execution Time: 1630ms
Thread-1 Execution Time: 4398ms
Program Execution Time: 4410ms
1...199999 prime# counter=17984
```

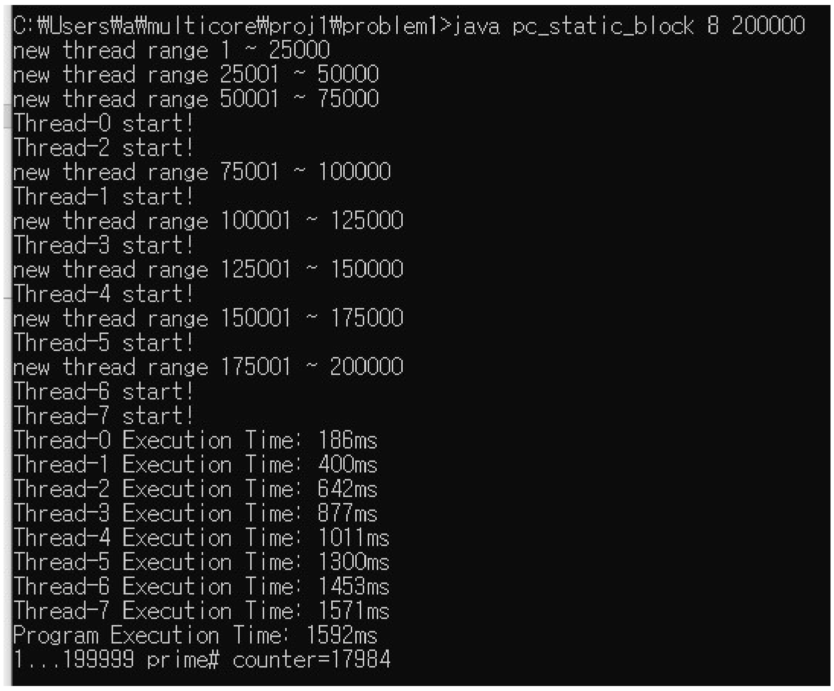
pc_static_block thread #4


```
C:\Users\a\multicore\proj1\problem1>java pc_static_block 4 200000
new thread range 1 ~ 50000
new thread range 50001 ~ 100000
new thread range 100001 ~ 150000
new thread range 150001 ~ 200000
Thread-2 start!
Thread-1 start!
Thread-0 start!
Thread-3 start!
Thread-0 Execution Time: 481ms
Thread-1 Execution Time: 1222ms
Thread-2 Execution Time: 1941ms
Thread-3 Execution Time: 2623ms
Program Execution Time: 2636ms
1...199999 prime# counter=17984
```

pc_static_block thread #6

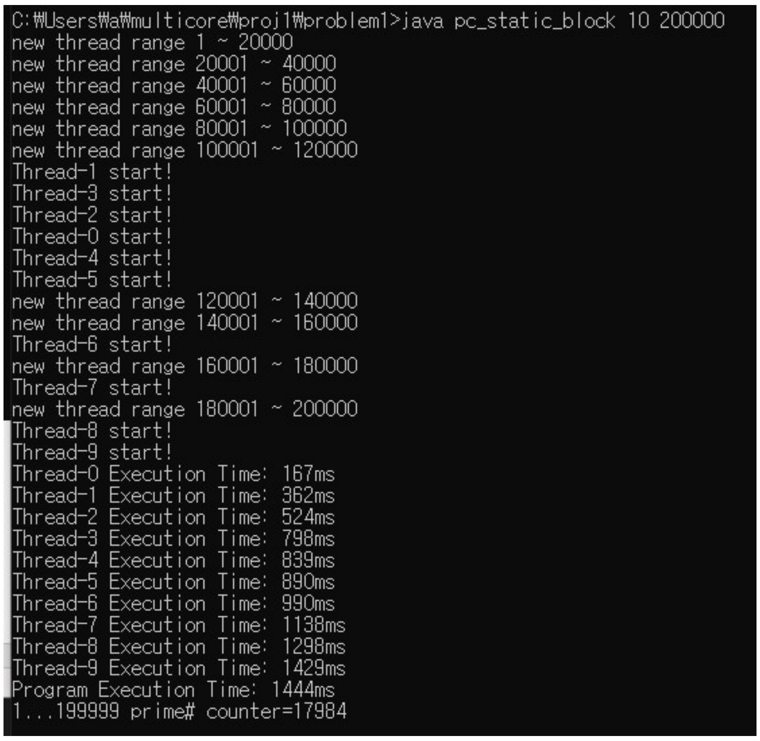

```
C:\Users\a\multicore\proj1\problem1>java pc_static_block 6 200000
new thread range 1 ~ 33333
new thread range 33334 ~ 66666
new thread range 66667 ~ 99999
new thread range 100000 ~ 133332
new thread range 133333 ~ 166665
new thread range 166666 ~ 200000
Thread-2 start!
Thread-3 start!
Thread-4 start!
Thread-0 start!
Thread-1 start!
Thread-5 start!
Thread-0 Execution Time: 252ms
Thread-1 Execution Time: 647ms
Thread-2 Execution Time: 963ms
Thread-3 Execution Time: 1296ms
Thread-4 Execution Time: 1629ms
Thread-5 Execution Time: 1890ms
Program Execution Time: 1904ms
1...199999 prime# counter=17984
```

pc_static_block thread #8



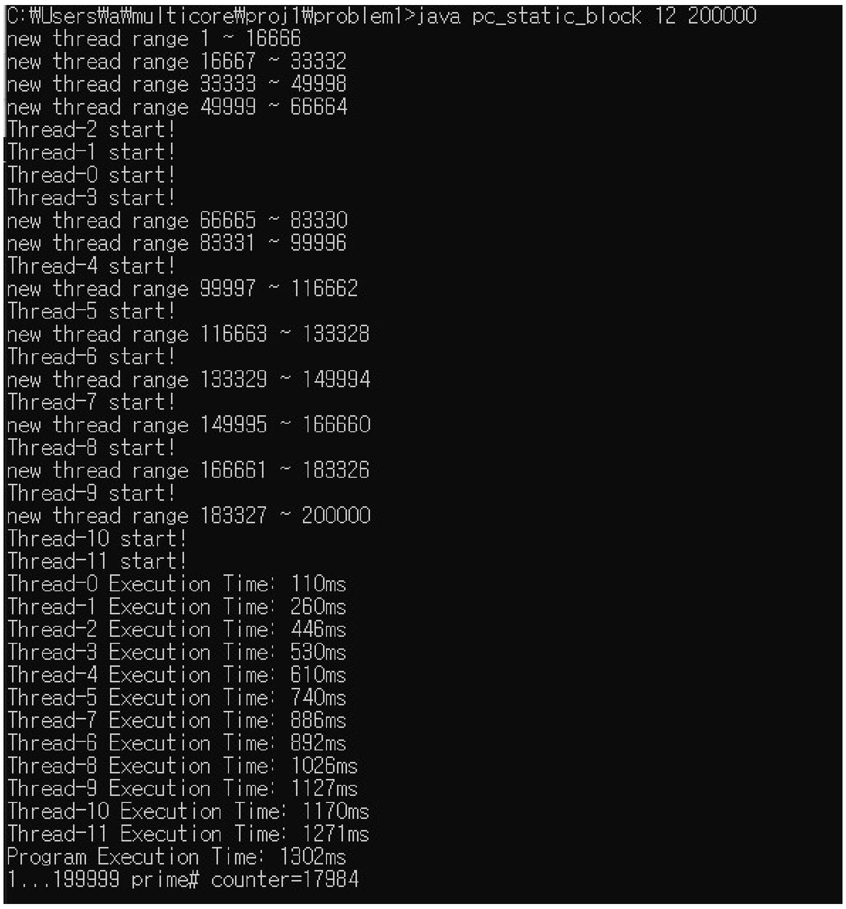
```
C:\Users\a\multicore\proj1\problem1>java pc_static_block 8 200000
new thread range 1 ~ 25000
new thread range 25001 ~ 50000
new thread range 50001 ~ 75000
Thread-0 start!
Thread-2 start!
new thread range 75001 ~ 100000
Thread-1 start!
new thread range 100001 ~ 125000
Thread-3 start!
new thread range 125001 ~ 150000
Thread-4 start!
new thread range 150001 ~ 175000
Thread-5 start!
new thread range 175001 ~ 200000
Thread-6 start!
Thread-7 start!
Thread-0 Execution Time: 186ms
Thread-1 Execution Time: 400ms
Thread-2 Execution Time: 642ms
Thread-3 Execution Time: 877ms
Thread-4 Execution Time: 1011ms
Thread-5 Execution Time: 1300ms
Thread-6 Execution Time: 1453ms
Thread-7 Execution Time: 1571ms
Program Execution Time: 1592ms
1...199999 prime# counter=17984
```

pc_static_block thread #10



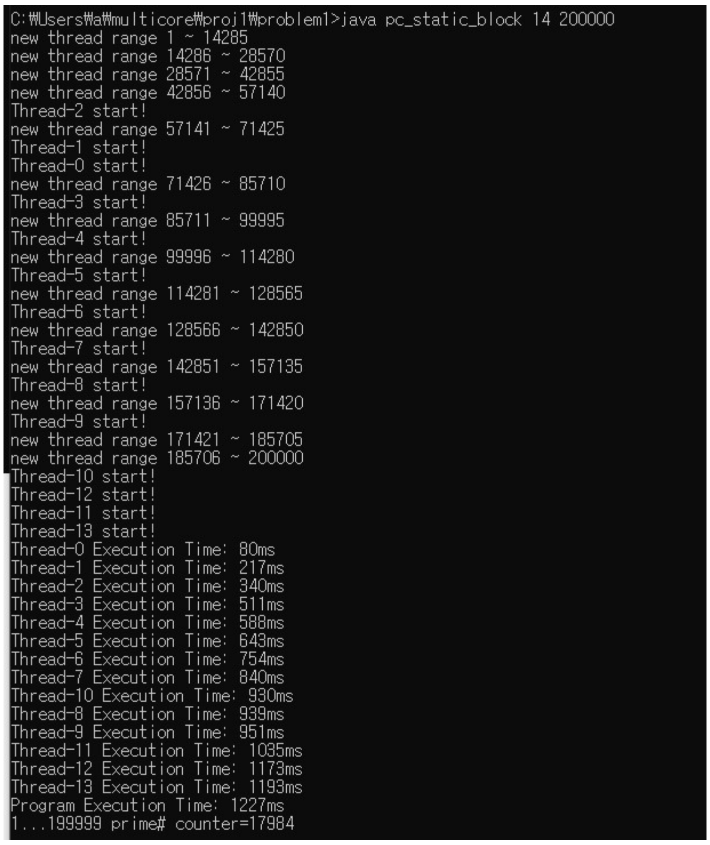
```
C:\Users\a\multicore\proj1\problem1>java pc_static_block 10 200000
new thread range 1 ~ 20000
new thread range 20001 ~ 40000
new thread range 40001 ~ 60000
new thread range 60001 ~ 80000
new thread range 80001 ~ 100000
new thread range 100001 ~ 120000
Thread-1 start!
Thread-3 start!
Thread-2 start!
Thread-0 start!
Thread-4 start!
Thread-5 start!
new thread range 120001 ~ 140000
new thread range 140001 ~ 160000
Thread-6 start!
new thread range 160001 ~ 180000
Thread-7 start!
new thread range 180001 ~ 200000
Thread-8 start!
Thread-9 start!
Thread-0 Execution Time: 167ms
Thread-1 Execution Time: 362ms
Thread-2 Execution Time: 524ms
Thread-3 Execution Time: 798ms
Thread-4 Execution Time: 839ms
Thread-5 Execution Time: 890ms
Thread-6 Execution Time: 990ms
Thread-7 Execution Time: 1138ms
Thread-8 Execution Time: 1298ms
Thread-9 Execution Time: 1429ms
Program Execution Time: 1444ms
1...199999 prime# counter=17984
```

pc_static_block thread #12



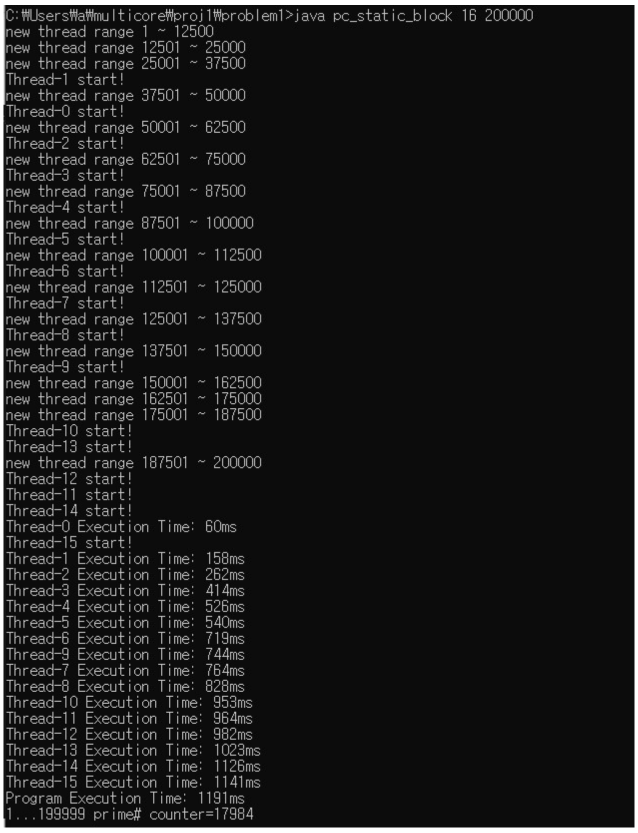
```
C:₩Users₩a₩multicore₩proj1₩problem1>java pc_static_block 12 200000
new thread range 1 ~ 16666
new thread range 16667 ~ 33332
new thread range 33333 ~ 49998
new thread range 49999 ~ 66664
Thread-2 start!
Thread-1 start!
Thread-0 start!
Thread-3 start!
new thread range 66665 ~ 83330
new thread range 83331 ~ 99996
Thread-4 start!
new thread range 99997 ~ 116662
Thread-5 start!
new thread range 116663 ~ 133328
Thread-6 start!
new thread range 133329 ~ 149994
Thread-7 start!
new thread range 149995 ~ 166660
Thread-8 start!
new thread range 166661 ~ 183326
Thread-9 start!
new thread range 183327 ~ 200000
Thread-10 start!
Thread-11 start!
Thread-0 Execution Time: 110ms
Thread-1 Execution Time: 260ms
Thread-2 Execution Time: 446ms
Thread-3 Execution Time: 530ms
Thread-4 Execution Time: 610ms
Thread-5 Execution Time: 740ms
Thread-7 Execution Time: 886ms
Thread-6 Execution Time: 892ms
Thread-8 Execution Time: 1026ms
Thread-9 Execution Time: 1127ms
Thread-10 Execution Time: 1170ms
Thread-11 Execution Time: 1271ms
Program Execution Time: 1302ms
1...199999 prime# counter=17984
```

pc_static_block thread #14



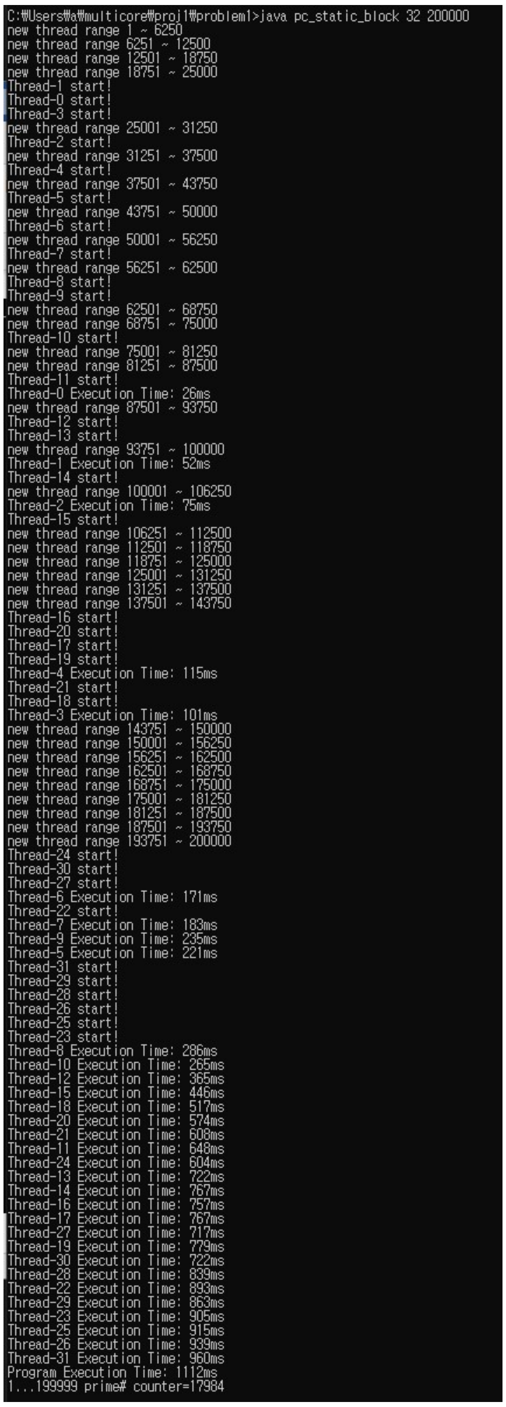
```
C:₩Users₩a₩multicore₩proj1₩problem1>java pc_static_block 14 200000
new thread range 1 ~ 14285
new thread range 14286 ~ 28570
new thread range 28571 ~ 42855
new thread range 42856 ~ 57140
Thread-2 start!
new thread range 57141 ~ 71425
Thread-1 start!
Thread-0 start!
new thread range 71426 ~ 85710
Thread-3 start!
new thread range 85711 ~ 99995
Thread-4 start!
new thread range 99996 ~ 114280
Thread-5 start!
new thread range 114281 ~ 128565
Thread-6 start!
new thread range 128566 ~ 142850
Thread-7 start!
new thread range 142851 ~ 157135
Thread-8 start!
new thread range 157136 ~ 171420
Thread-9 start!
new thread range 171421 ~ 185705
new thread range 185706 ~ 200000
Thread-10 start!
Thread-12 start!
Thread-11 start!
Thread-13 start!
Thread-0 Execution Time: 80ms
Thread-1 Execution Time: 217ms
Thread-2 Execution Time: 340ms
Thread-3 Execution Time: 511ms
Thread-4 Execution Time: 588ms
Thread-5 Execution Time: 643ms
Thread-6 Execution Time: 754ms
Thread-7 Execution Time: 840ms
Thread-10 Execution Time: 930ms
Thread-8 Execution Time: 939ms
Thread-9 Execution Time: 951ms
Thread-11 Execution Time: 1035ms
Thread-12 Execution Time: 1173ms
Thread-13 Execution Time: 1193ms
Program Execution Time: 1227ms
1...199999 prime# counter=17984
```

pc_static_block thread #16



```
C:₩Users₩a₩multicore₩proj1₩problem1>java pc_static_block 16 200000
new thread range 1 ~ 12500
new thread range 12501 ~ 25000
new thread range 25001 ~ 37500
Thread-1 start!
new thread range 37501 ~ 50000
Thread-0 start!
new thread range 50001 ~ 62500
Thread-2 start!
new thread range 62501 ~ 75000
Thread-3 start!
new thread range 75001 ~ 87500
Thread-4 start!
new thread range 87501 ~ 100000
Thread-5 start!
new thread range 100001 ~ 112500
Thread-6 start!
new thread range 112501 ~ 125000
Thread-7 start!
new thread range 125001 ~ 137500
Thread-8 start!
new thread range 137501 ~ 150000
Thread-9 start!
new thread range 150001 ~ 162500
new thread range 162501 ~ 175000
new thread range 175001 ~ 187500
Thread-10 start!
Thread-13 start!
new thread range 187501 ~ 200000
Thread-12 start!
Thread-11 start!
Thread-14 start!
Thread-0 Execution Time: 60ms
Thread-15 start!
Thread-1 Execution Time: 158ms
Thread-2 Execution Time: 262ms
Thread-3 Execution Time: 414ms
Thread-4 Execution Time: 526ms
Thread-5 Execution Time: 540ms
Thread-6 Execution Time: 719ms
Thread-9 Execution Time: 744ms
Thread-7 Execution Time: 764ms
Thread-8 Execution Time: 828ms
Thread-10 Execution Time: 953ms
Thread-11 Execution Time: 964ms
Thread-12 Execution Time: 982ms
Thread-13 Execution Time: 1023ms
Thread-14 Execution Time: 1126ms
Thread-15 Execution Time: 1141ms
Program Execution Time: 1191ms
1...199999 prime# counter=17984
```

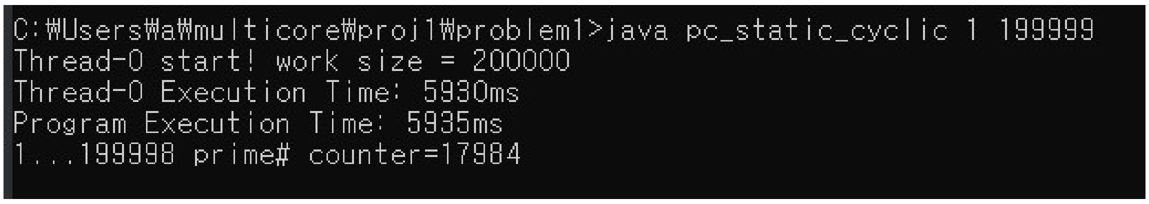
pc_static_block thread #32



```
C:₩Users₩a₩multicore₩proj1₩problem1>java pc_static_block 32 200000
new thread range 1 ~ 6250
new thread range 6251 ~ 12500
new thread range 12501 ~ 18750
new thread range 18751 ~ 25000
Thread-1 start!
Thread-0 start!
Thread-3 start!
new thread range 25001 ~ 31250
Thread-2 start!
new thread range 31251 ~ 37500
Thread-4 start!
new thread range 37501 ~ 43750
Thread-5 start!
new thread range 43751 ~ 50000
Thread-6 start!
new thread range 50001 ~ 56250
Thread-7 start!
new thread range 56251 ~ 62500
Thread-8 start!
Thread-9 start!
new thread range 62501 ~ 68750
new thread range 68751 ~ 75000
Thread-10 start!
new thread range 75001 ~ 81250
new thread range 81251 ~ 87500
Thread-11 start!
Thread-0 Execution Time: 26ms
new thread range 87501 ~ 93750
Thread-12 start!
Thread-13 start!
new thread range 93751 ~ 100000
Thread-1 Execution Time: 52ms
Thread-14 start!
new thread range 100001 ~ 106250
Thread-2 Execution Time: 75ms
Thread-15 start!
new thread range 106251 ~ 112500
new thread range 112501 ~ 118750
new thread range 118751 ~ 125000
new thread range 125001 ~ 131250
new thread range 131251 ~ 137500
new thread range 137501 ~ 143750
Thread-16 start!
Thread-20 start!
Thread-17 start!
Thread-19 start!
Thread-4 Execution Time: 115ms
Thread-21 start!
Thread-18 start!
Thread-3 Execution Time: 101ms
new thread range 143751 ~ 150000
new thread range 150001 ~ 156250
new thread range 156251 ~ 162500
new thread range 162501 ~ 168750
new thread range 168751 ~ 175000
new thread range 175001 ~ 181250
new thread range 181251 ~ 187500
new thread range 187501 ~ 193750
new thread range 193751 ~ 200000
Thread-24 start!
Thread-30 start!
Thread-27 start!
Thread-6 Execution Time: 171ms
Thread-22 start!
Thread-7 Execution Time: 183ms
Thread-9 Execution Time: 235ms
Thread-5 Execution Time: 221ms
Thread-31 start!
Thread-29 start!
Thread-28 start!
Thread-26 start!
Thread-25 start!
Thread-23 start!
Thread-8 Execution Time: 286ms
Thread-10 Execution Time: 265ms
Thread-12 Execution Time: 365ms
Thread-15 Execution Time: 446ms
Thread-18 Execution Time: 517ms
Thread-20 Execution Time: 574ms
Thread-21 Execution Time: 608ms
Thread-11 Execution Time: 648ms
Thread-24 Execution Time: 604ms
Thread-13 Execution Time: 722ms
Thread-14 Execution Time: 767ms
Thread-16 Execution Time: 757ms
Thread-17 Execution Time: 767ms
Thread-27 Execution Time: 717ms
Thread-19 Execution Time: 779ms
Thread-30 Execution Time: 722ms
Thread-28 Execution Time: 839ms
Thread-22 Execution Time: 893ms
Thread-29 Execution Time: 863ms
Thread-23 Execution Time: 905ms
Thread-25 Execution Time: 915ms
Thread-26 Execution Time: 939ms
Thread-31 Execution Time: 960ms
Program Execution Time: 1112ms
1...199999 prime# counter=17984
```

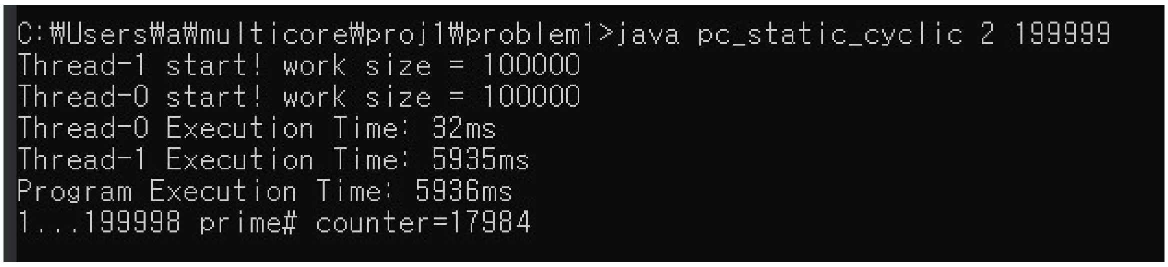
PC_static_cyclic.java

pc_static_cyclic thread #1



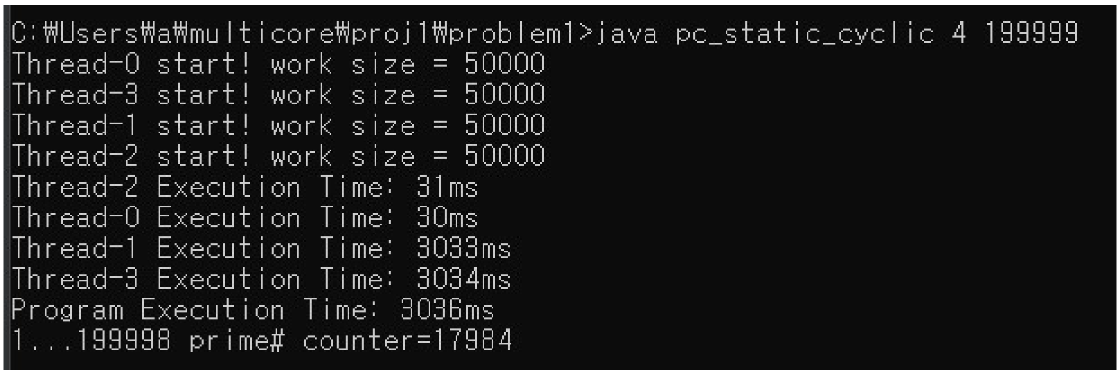
```
C:₩Users₩a₩multicore₩proj1₩problem1>java pc_static_cyclic 1 199999
Thread-0 start! work size = 200000
Thread-0 Execution Time: 5930ms
Program Execution Time: 5935ms
1...199998 prime# counter=17984
```

pc_static_cyclic thread #2



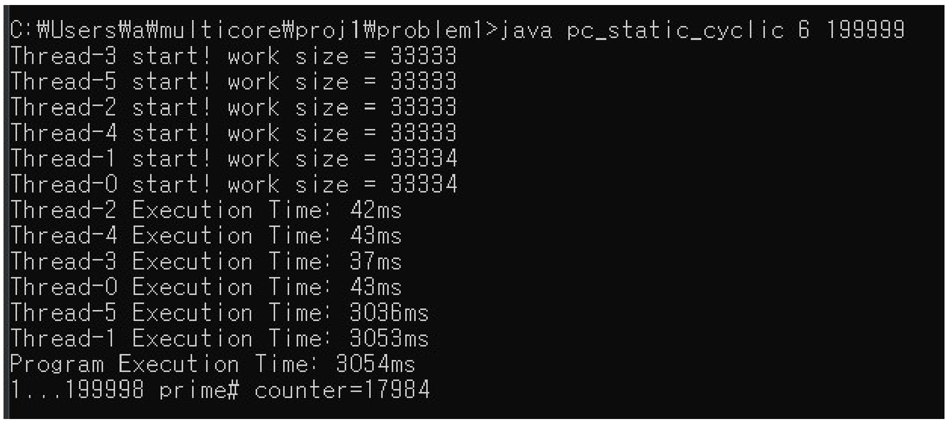
```
C:₩Users₩a₩multicore₩proj1₩problem1>java pc_static_cyclic 2 199999
Thread-1 start! work size = 100000
Thread-0 start! work size = 100000
Thread-0 Execution Time: 32ms
Thread-1 Execution Time: 5935ms
Program Execution Time: 5936ms
1...199998 prime# counter=17984
```

pc_static_cyclic thread #4



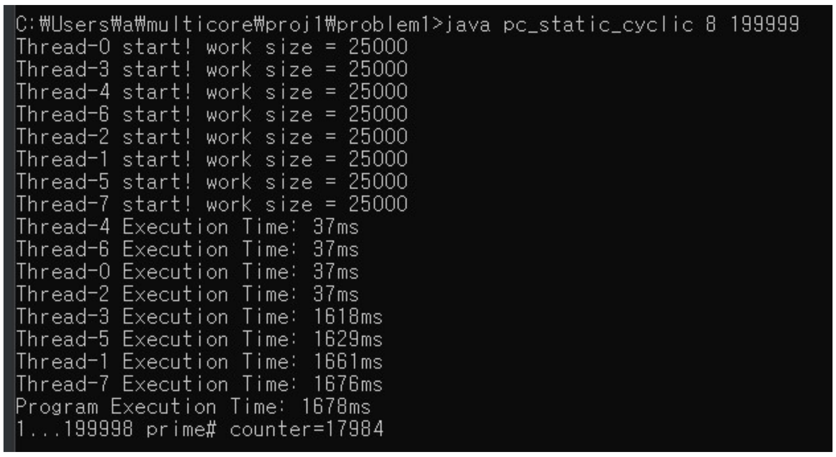
```
C:₩Users₩a₩multicore₩proj1₩problem1>java pc_static_cyclic 4 199999
Thread-0 start! work size = 50000
Thread-3 start! work size = 50000
Thread-1 start! work size = 50000
Thread-2 start! work size = 50000
Thread-2 Execution Time: 31ms
Thread-0 Execution Time: 30ms
Thread-1 Execution Time: 3033ms
Thread-3 Execution Time: 3034ms
Program Execution Time: 3036ms
1...199998 prime# counter=17984
```

pc_static_cyclic thread #6



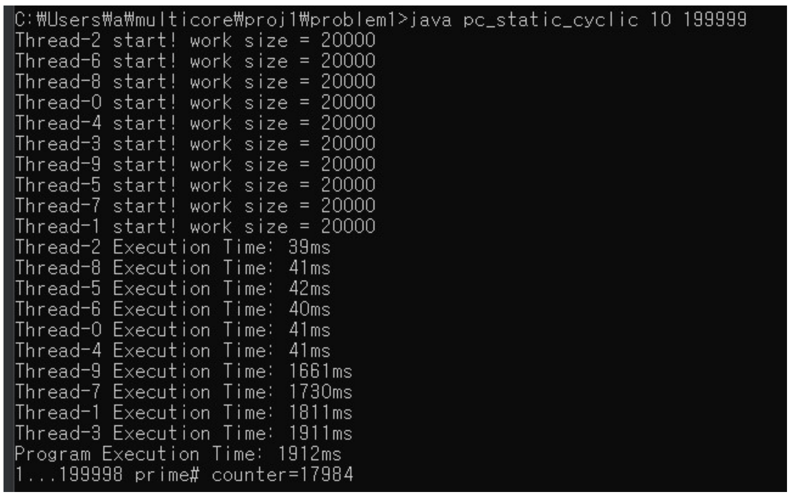
```
C:₩Users₩a₩multicore₩proj1₩problem1>java pc_static_cyclic 6 199999
Thread-3 start! work size = 33333
Thread-5 start! work size = 33333
Thread-2 start! work size = 33333
Thread-4 start! work size = 33333
Thread-1 start! work size = 33334
Thread-0 start! work size = 33334
Thread-2 Execution Time: 42ms
Thread-4 Execution Time: 43ms
Thread-3 Execution Time: 37ms
Thread-0 Execution Time: 43ms
Thread-5 Execution Time: 3036ms
Thread-1 Execution Time: 3053ms
Program Execution Time: 3054ms
1...199998 prime# counter=17984
```

pc_static_cyclic thread #8



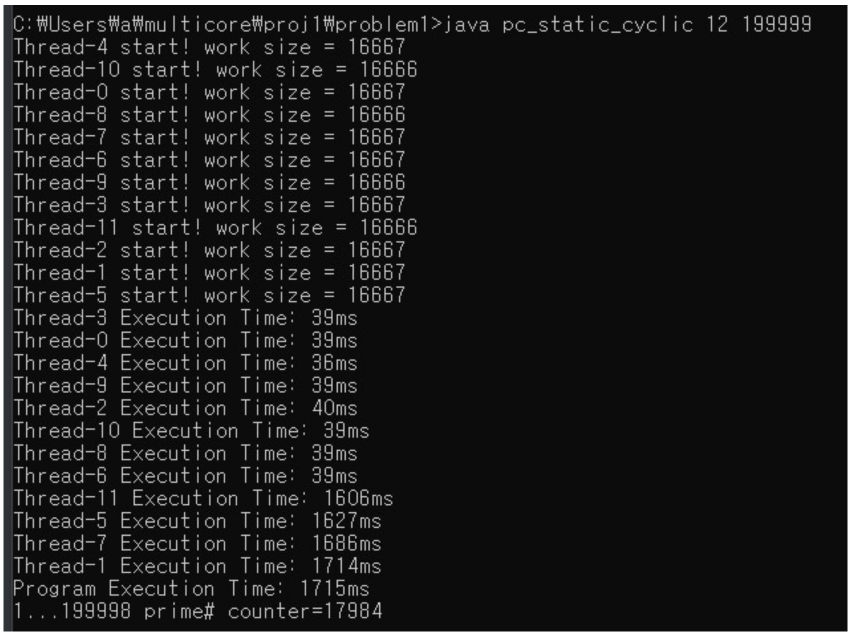
```
C:\Users\a\multicore\proj1\problem1>java pc_static_cyclic 8 199999
Thread-0 start! work size = 25000
Thread-3 start! work size = 25000
Thread-4 start! work size = 25000
Thread-6 start! work size = 25000
Thread-2 start! work size = 25000
Thread-1 start! work size = 25000
Thread-5 start! work size = 25000
Thread-7 start! work size = 25000
Thread-4 Execution Time: 37ms
Thread-6 Execution Time: 37ms
Thread-0 Execution Time: 37ms
Thread-2 Execution Time: 37ms
Thread-3 Execution Time: 1618ms
Thread-5 Execution Time: 1629ms
Thread-1 Execution Time: 1661ms
Thread-7 Execution Time: 1676ms
Program Execution Time: 1678ms
1...199998 prime# counter=17984
```

pc_static_cyclic thread #10



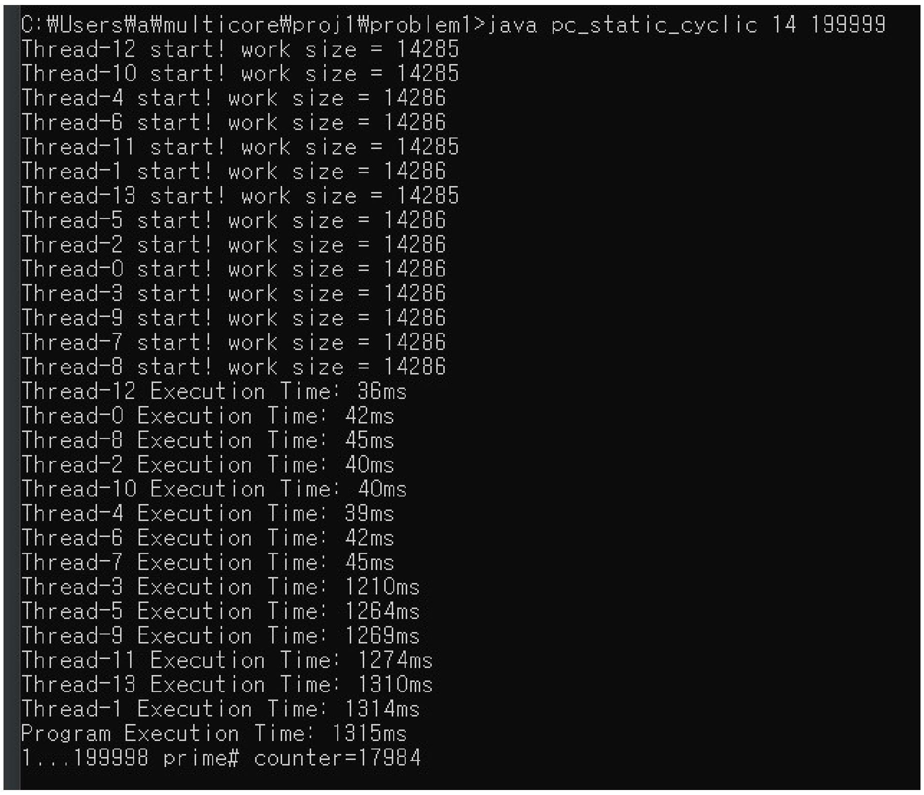
```
C:\Users\a\multicore\proj1\problem1>java pc_static_cyclic 10 199999
Thread-2 start! work size = 20000
Thread-6 start! work size = 20000
Thread-8 start! work size = 20000
Thread-0 start! work size = 20000
Thread-4 start! work size = 20000
Thread-3 start! work size = 20000
Thread-9 start! work size = 20000
Thread-5 start! work size = 20000
Thread-7 start! work size = 20000
Thread-1 start! work size = 20000
Thread-2 Execution Time: 39ms
Thread-8 Execution Time: 41ms
Thread-5 Execution Time: 42ms
Thread-6 Execution Time: 40ms
Thread-0 Execution Time: 41ms
Thread-4 Execution Time: 41ms
Thread-9 Execution Time: 1661ms
Thread-7 Execution Time: 1730ms
Thread-1 Execution Time: 1811ms
Thread-3 Execution Time: 1911ms
Program Execution Time: 1912ms
1...199998 prime# counter=17984
```

pc_static_cyclic thread #12



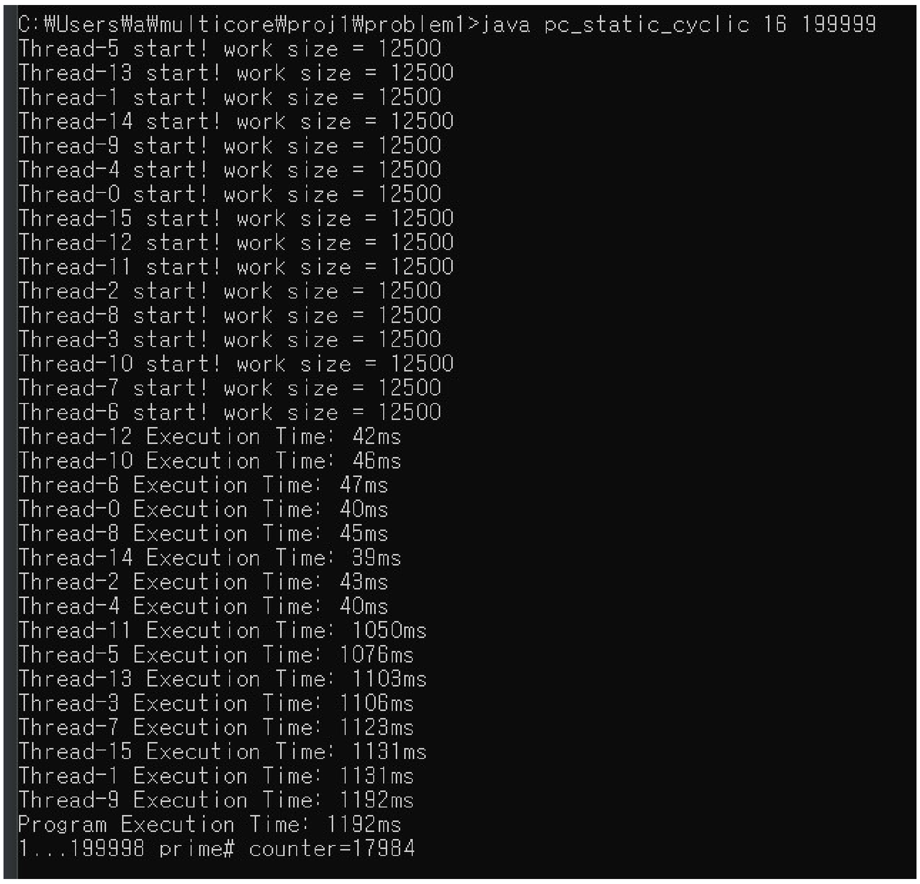
```
C:\Users\a\multicore\proj1\problem1>java pc_static_cyclic 12 199999
Thread-4 start! work size = 16667
Thread-10 start! work size = 16666
Thread-0 start! work size = 16667
Thread-8 start! work size = 16666
Thread-7 start! work size = 16667
Thread-6 start! work size = 16667
Thread-9 start! work size = 16666
Thread-3 start! work size = 16667
Thread-11 start! work size = 16666
Thread-2 start! work size = 16667
Thread-1 start! work size = 16667
Thread-5 start! work size = 16667
Thread-3 Execution Time: 39ms
Thread-0 Execution Time: 39ms
Thread-4 Execution Time: 36ms
Thread-9 Execution Time: 39ms
Thread-2 Execution Time: 40ms
Thread-10 Execution Time: 39ms
Thread-8 Execution Time: 39ms
Thread-6 Execution Time: 39ms
Thread-11 Execution Time: 1606ms
Thread-5 Execution Time: 1627ms
Thread-7 Execution Time: 1686ms
Thread-1 Execution Time: 1714ms
Program Execution Time: 1715ms
1...199998 prime# counter=17984
```

pc_static_cyclic thread #14



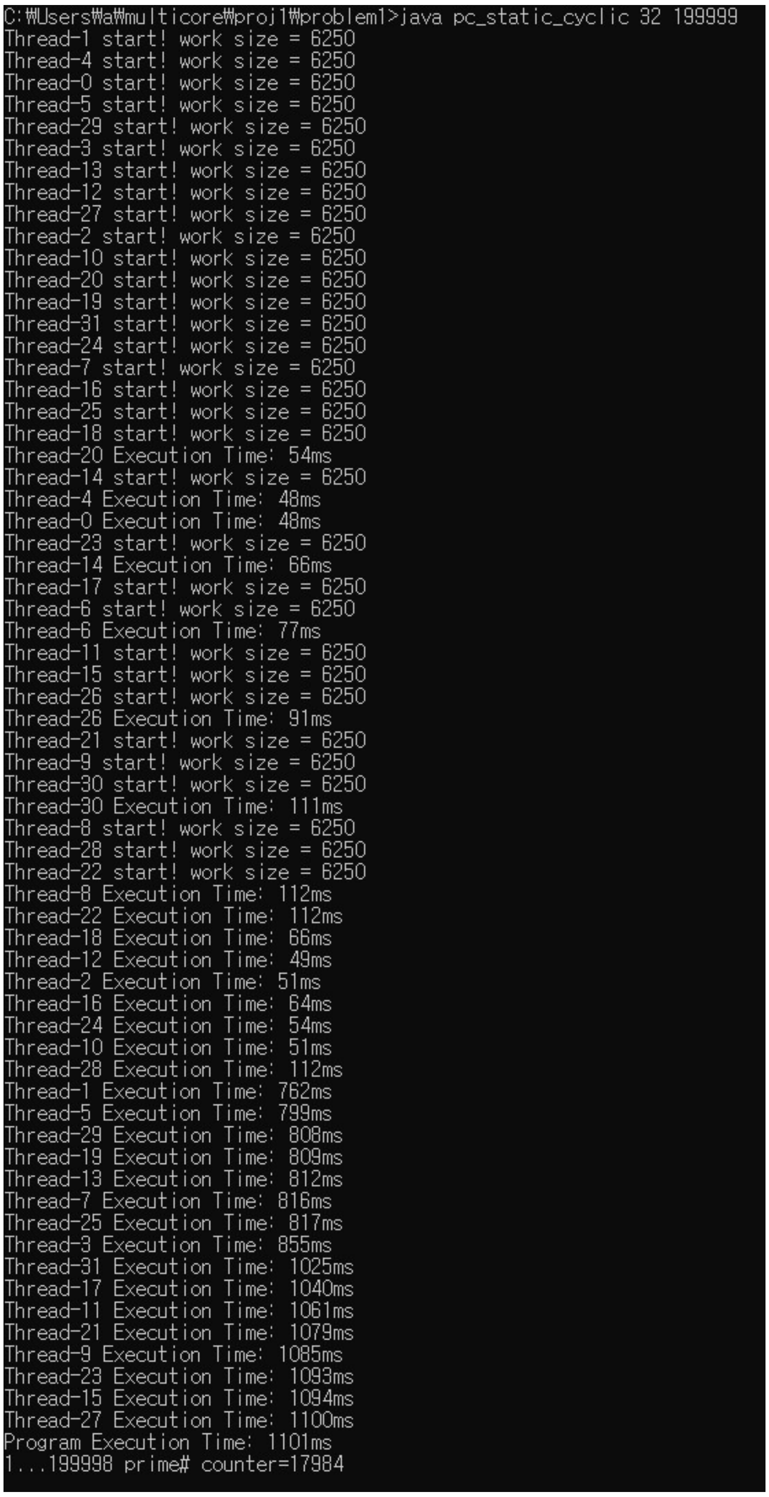
```
C:₩Users₩a₩multicore₩proj1₩problem1>java pc_static_cyclic 14 199999
Thread-12 start! work size = 14285
Thread-10 start! work size = 14285
Thread-4 start! work size = 14286
Thread-6 start! work size = 14286
Thread-11 start! work size = 14285
Thread-1 start! work size = 14286
Thread-13 start! work size = 14285
Thread-5 start! work size = 14286
Thread-2 start! work size = 14286
Thread-0 start! work size = 14286
Thread-3 start! work size = 14286
Thread-9 start! work size = 14286
Thread-7 start! work size = 14286
Thread-8 start! work size = 14286
Thread-12 Execution Time: 36ms
Thread-0 Execution Time: 42ms
Thread-8 Execution Time: 45ms
Thread-2 Execution Time: 40ms
Thread-10 Execution Time: 40ms
Thread-4 Execution Time: 39ms
Thread-6 Execution Time: 42ms
Thread-7 Execution Time: 45ms
Thread-3 Execution Time: 1210ms
Thread-5 Execution Time: 1264ms
Thread-9 Execution Time: 1269ms
Thread-11 Execution Time: 1274ms
Thread-13 Execution Time: 1310ms
Thread-1 Execution Time: 1314ms
Program Execution Time: 1315ms
1...199998 prime# counter=17984
```

pc_static_cyclic thread #16



```
C:₩Users₩a₩multicore₩proj1₩problem1>java pc_static_cyclic 16 199999
Thread-5 start! work size = 12500
Thread-13 start! work size = 12500
Thread-1 start! work size = 12500
Thread-14 start! work size = 12500
Thread-9 start! work size = 12500
Thread-4 start! work size = 12500
Thread-0 start! work size = 12500
Thread-15 start! work size = 12500
Thread-12 start! work size = 12500
Thread-11 start! work size = 12500
Thread-2 start! work size = 12500
Thread-8 start! work size = 12500
Thread-3 start! work size = 12500
Thread-10 start! work size = 12500
Thread-7 start! work size = 12500
Thread-6 start! work size = 12500
Thread-12 Execution Time: 42ms
Thread-10 Execution Time: 46ms
Thread-6 Execution Time: 47ms
Thread-0 Execution Time: 40ms
Thread-8 Execution Time: 45ms
Thread-14 Execution Time: 39ms
Thread-2 Execution Time: 43ms
Thread-4 Execution Time: 40ms
Thread-11 Execution Time: 1050ms
Thread-5 Execution Time: 1076ms
Thread-13 Execution Time: 1103ms
Thread-3 Execution Time: 1106ms
Thread-7 Execution Time: 1123ms
Thread-15 Execution Time: 1131ms
Thread-1 Execution Time: 1131ms
Thread-9 Execution Time: 1192ms
Program Execution Time: 1192ms
1...199998 prime# counter=17984
```

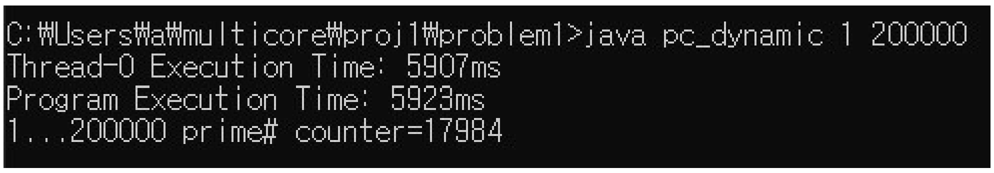
pc_static_cyclic thread #32



```
C:\Users\a\multicore\proj1\problem1>java pc_static_cyclic 32 199999
Thread-1 start! work size = 6250
Thread-4 start! work size = 6250
Thread-0 start! work size = 6250
Thread-5 start! work size = 6250
Thread-29 start! work size = 6250
Thread-3 start! work size = 6250
Thread-13 start! work size = 6250
Thread-12 start! work size = 6250
Thread-27 start! work size = 6250
Thread-2 start! work size = 6250
Thread-10 start! work size = 6250
Thread-20 start! work size = 6250
Thread-19 start! work size = 6250
Thread-31 start! work size = 6250
Thread-24 start! work size = 6250
Thread-7 start! work size = 6250
Thread-16 start! work size = 6250
Thread-25 start! work size = 6250
Thread-18 start! work size = 6250
Thread-20 Execution Time: 54ms
Thread-14 start! work size = 6250
Thread-4 Execution Time: 48ms
Thread-0 Execution Time: 48ms
Thread-23 start! work size = 6250
Thread-14 Execution Time: 66ms
Thread-17 start! work size = 6250
Thread-6 start! work size = 6250
Thread-6 Execution Time: 77ms
Thread-11 start! work size = 6250
Thread-15 start! work size = 6250
Thread-26 start! work size = 6250
Thread-26 Execution Time: 91ms
Thread-21 start! work size = 6250
Thread-9 start! work size = 6250
Thread-30 start! work size = 6250
Thread-30 Execution Time: 111ms
Thread-8 start! work size = 6250
Thread-28 start! work size = 6250
Thread-22 start! work size = 6250
Thread-8 Execution Time: 112ms
Thread-22 Execution Time: 112ms
Thread-18 Execution Time: 66ms
Thread-12 Execution Time: 49ms
Thread-2 Execution Time: 51ms
Thread-16 Execution Time: 64ms
Thread-24 Execution Time: 54ms
Thread-10 Execution Time: 51ms
Thread-28 Execution Time: 112ms
Thread-1 Execution Time: 762ms
Thread-5 Execution Time: 799ms
Thread-29 Execution Time: 808ms
Thread-19 Execution Time: 809ms
Thread-13 Execution Time: 812ms
Thread-7 Execution Time: 816ms
Thread-25 Execution Time: 817ms
Thread-3 Execution Time: 855ms
Thread-31 Execution Time: 1025ms
Thread-17 Execution Time: 1040ms
Thread-11 Execution Time: 1061ms
Thread-21 Execution Time: 1079ms
Thread-9 Execution Time: 1085ms
Thread-23 Execution Time: 1093ms
Thread-15 Execution Time: 1094ms
Thread-27 Execution Time: 1100ms
Program Execution Time: 1101ms
1...199998 prime# counter=17984
```

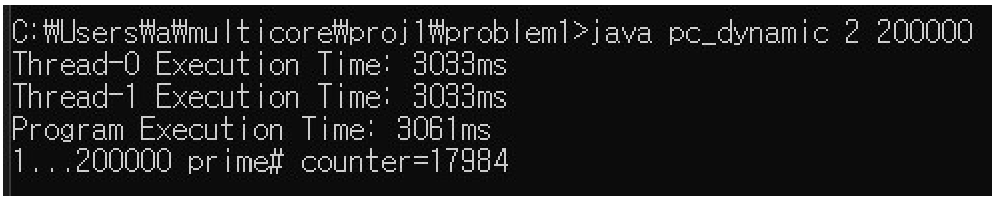
pc_dynamic.java

pc_dynamic thread #1



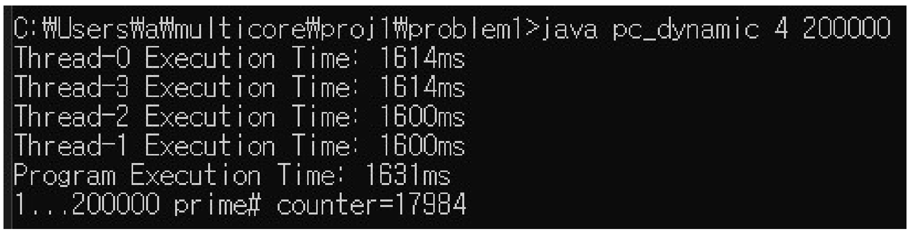
```
C:₩Users₩a₩multicore₩proj1₩problem1>java pc_dynamic 1 200000
Thread-0 Execution Time: 5907ms
Program Execution Time: 5923ms
1...200000 prime# counter=17984
```

pc_dynamic thread #2



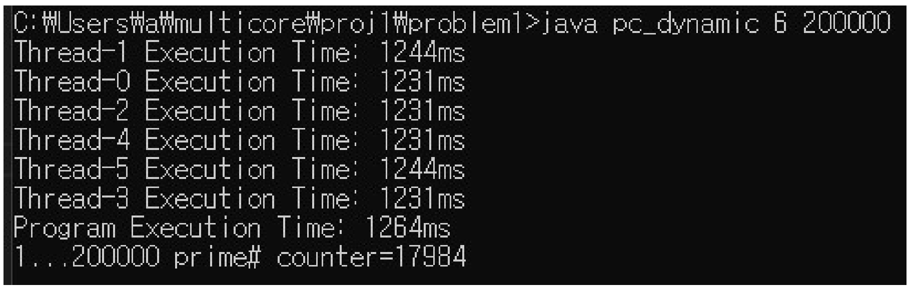
```
C:₩Users₩a₩multicore₩proj1₩problem1>java pc_dynamic 2 200000
Thread-0 Execution Time: 3033ms
Thread-1 Execution Time: 3033ms
Program Execution Time: 3061ms
1...200000 prime# counter=17984
```

pc_dynamic thread #4



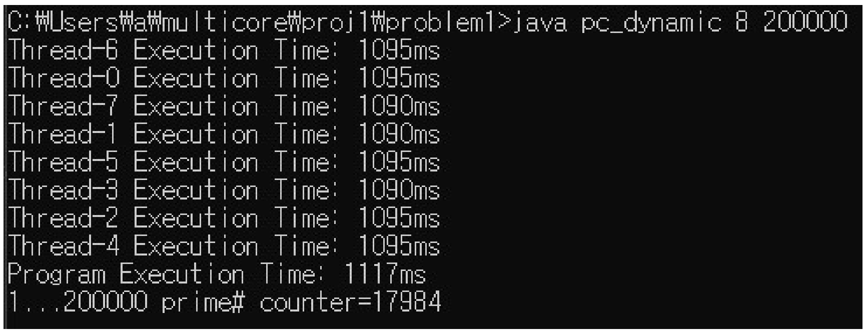
```
C:₩Users₩a₩multicore₩proj1₩problem1>java pc_dynamic 4 200000
Thread-0 Execution Time: 1614ms
Thread-3 Execution Time: 1614ms
Thread-2 Execution Time: 1600ms
Thread-1 Execution Time: 1600ms
Program Execution Time: 1631ms
1...200000 prime# counter=17984
```

pc_dynamic thread #6



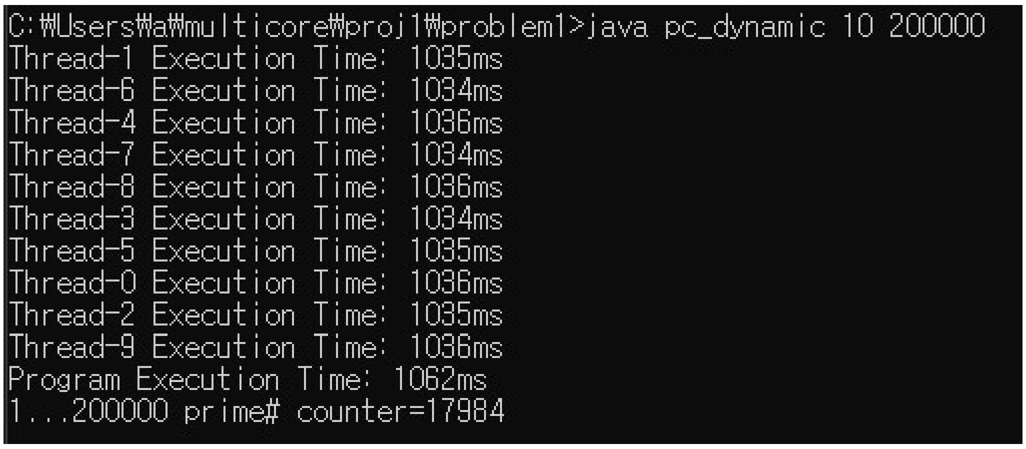
```
C:₩Users₩a₩multicore₩proj1₩problem1>java pc_dynamic 6 200000
Thread-1 Execution Time: 1244ms
Thread-0 Execution Time: 1231ms
Thread-2 Execution Time: 1231ms
Thread-4 Execution Time: 1231ms
Thread-5 Execution Time: 1244ms
Thread-3 Execution Time: 1231ms
Program Execution Time: 1264ms
1...200000 prime# counter=17984
```

pc_dynamic thread #8



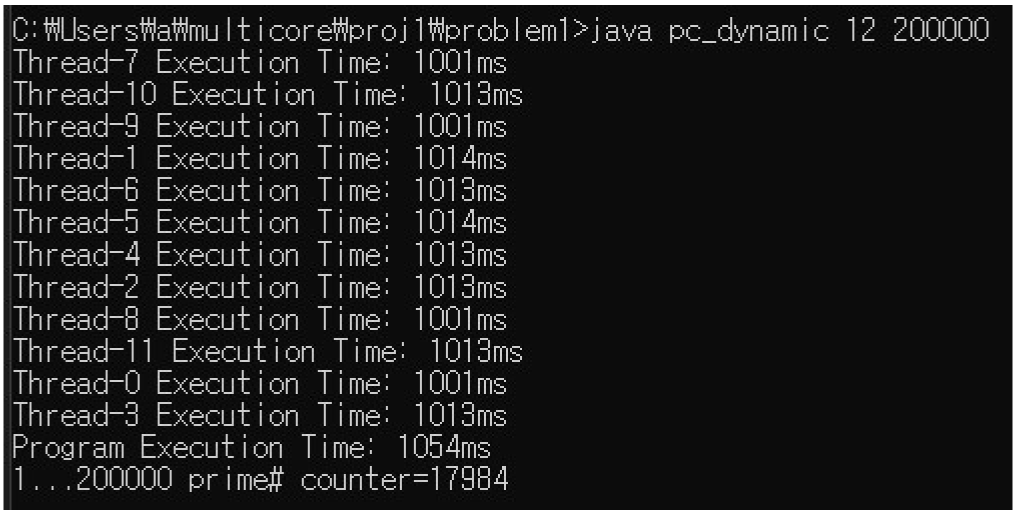
```
C:₩Users₩a₩multicore₩proj1₩problem1>java pc_dynamic 8 200000
Thread-6 Execution Time: 1095ms
Thread-0 Execution Time: 1095ms
Thread-7 Execution Time: 1090ms
Thread-1 Execution Time: 1090ms
Thread-5 Execution Time: 1095ms
Thread-3 Execution Time: 1090ms
Thread-2 Execution Time: 1095ms
Thread-4 Execution Time: 1095ms
Program Execution Time: 1117ms
1...200000 prime# counter=17984
```

pc_dynamic thread #10



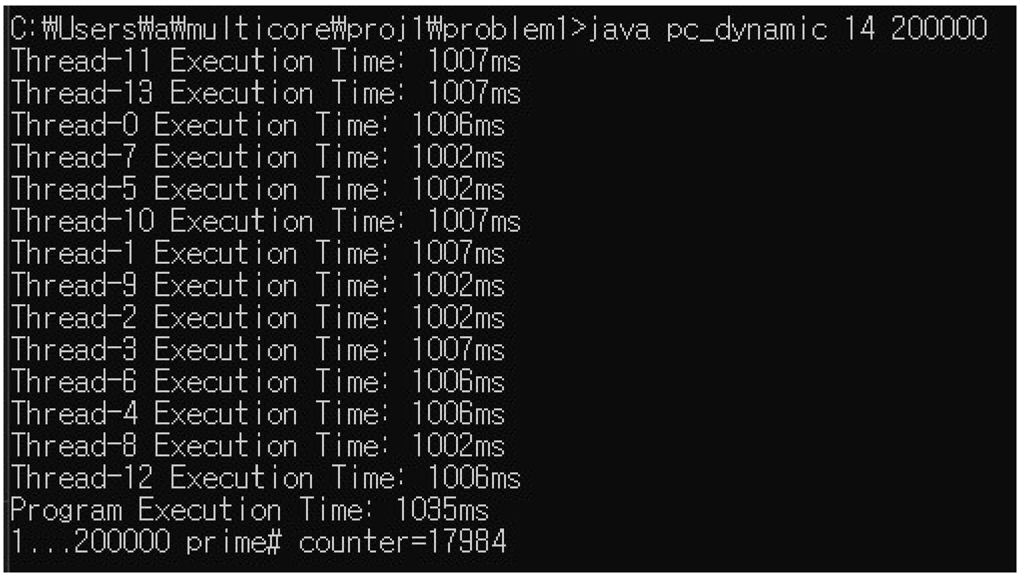
```
C:\Users\a\multicore\proj1\problem1>java pc_dynamic 10 200000
Thread-1 Execution Time: 1035ms
Thread-6 Execution Time: 1034ms
Thread-4 Execution Time: 1036ms
Thread-7 Execution Time: 1034ms
Thread-8 Execution Time: 1036ms
Thread-3 Execution Time: 1034ms
Thread-5 Execution Time: 1035ms
Thread-0 Execution Time: 1036ms
Thread-2 Execution Time: 1035ms
Thread-9 Execution Time: 1036ms
Program Execution Time: 1062ms
1...200000 prime# counter=17984
```

pc_dynamic thread #12



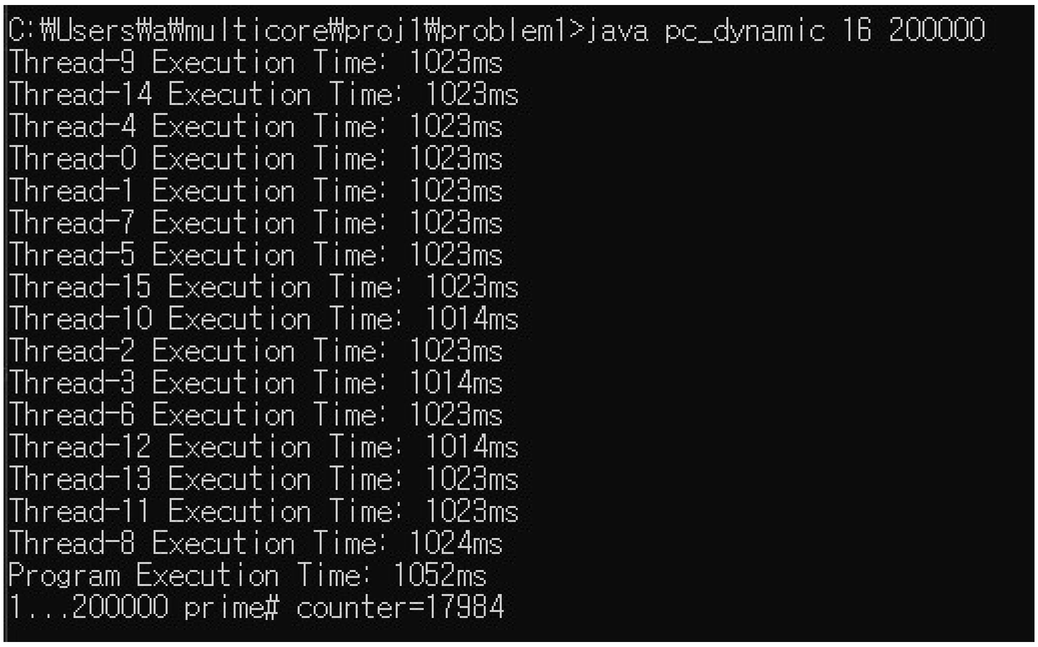
```
C:\Users\a\multicore\proj1\problem1>java pc_dynamic 12 200000
Thread-7 Execution Time: 1001ms
Thread-10 Execution Time: 1013ms
Thread-9 Execution Time: 1001ms
Thread-1 Execution Time: 1014ms
Thread-6 Execution Time: 1013ms
Thread-5 Execution Time: 1014ms
Thread-4 Execution Time: 1013ms
Thread-2 Execution Time: 1013ms
Thread-8 Execution Time: 1001ms
Thread-11 Execution Time: 1013ms
Thread-0 Execution Time: 1001ms
Thread-3 Execution Time: 1013ms
Program Execution Time: 1054ms
1...200000 prime# counter=17984
```

pc_dynamic thread #14

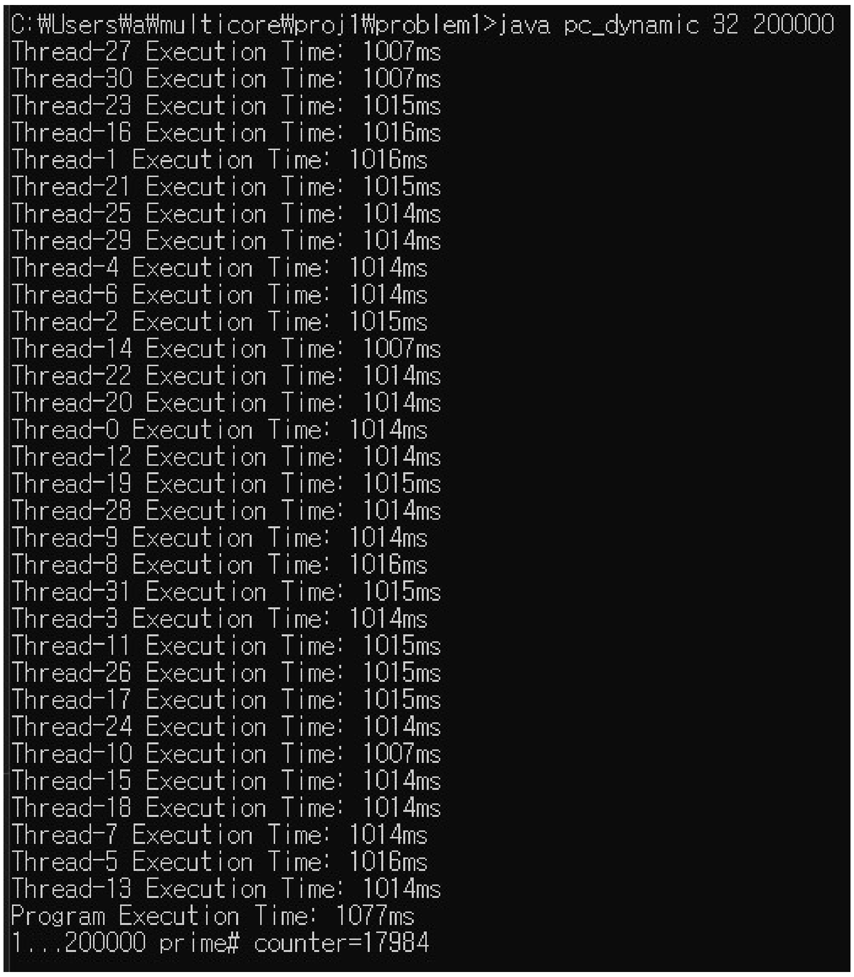


```
C:\Users\a\multicore\proj1\problem1>java pc_dynamic 14 200000
Thread-11 Execution Time: 1007ms
Thread-13 Execution Time: 1007ms
Thread-0 Execution Time: 1006ms
Thread-7 Execution Time: 1002ms
Thread-5 Execution Time: 1002ms
Thread-10 Execution Time: 1007ms
Thread-1 Execution Time: 1007ms
Thread-9 Execution Time: 1002ms
Thread-2 Execution Time: 1002ms
Thread-3 Execution Time: 1007ms
Thread-6 Execution Time: 1006ms
Thread-4 Execution Time: 1006ms
Thread-8 Execution Time: 1002ms
Thread-12 Execution Time: 1006ms
Program Execution Time: 1035ms
1...200000 prime# counter=17984
```

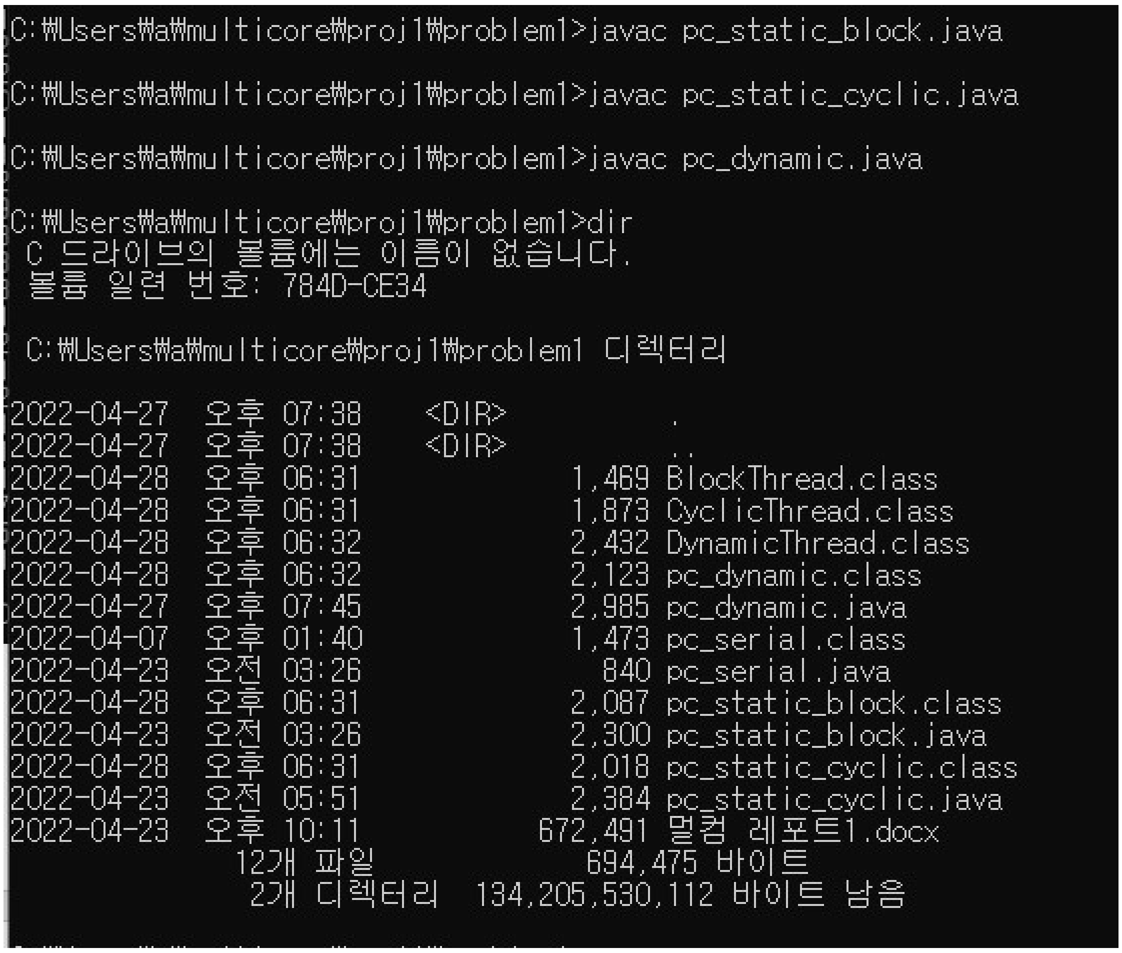
pc_dynamic thread #16


```
C:₩Users₩a₩multicore₩proj1₩problem1>java pc_dynamic 16 200000
Thread-9 Execution Time: 1023ms
Thread-14 Execution Time: 1023ms
Thread-4 Execution Time: 1023ms
Thread-0 Execution Time: 1023ms
Thread-1 Execution Time: 1023ms
Thread-7 Execution Time: 1023ms
Thread-5 Execution Time: 1023ms
Thread-15 Execution Time: 1023ms
Thread-10 Execution Time: 1014ms
Thread-2 Execution Time: 1023ms
Thread-3 Execution Time: 1014ms
Thread-6 Execution Time: 1023ms
Thread-12 Execution Time: 1014ms
Thread-13 Execution Time: 1023ms
Thread-11 Execution Time: 1023ms
Thread-8 Execution Time: 1024ms
Program Execution Time: 1052ms
1...200000 prime# counter=17984
```

pc_dynamic thread #32

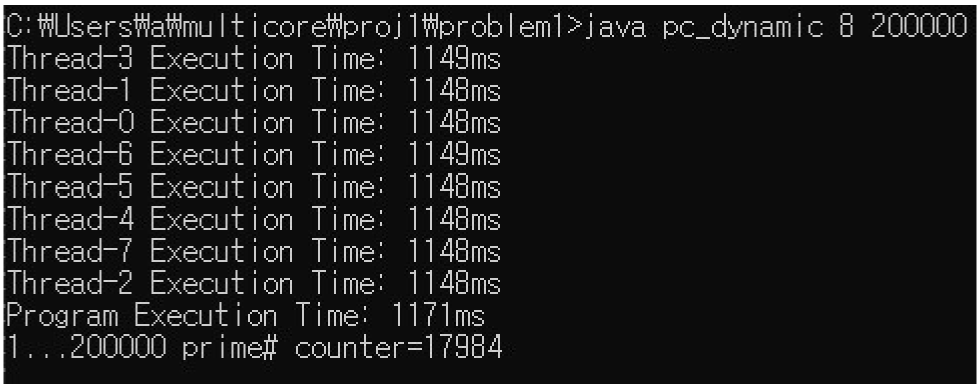

```
C:₩Users₩a₩multicore₩proj1₩problem1>java pc_dynamic 32 200000
Thread-27 Execution Time: 1007ms
Thread-30 Execution Time: 1007ms
Thread-23 Execution Time: 1015ms
Thread-16 Execution Time: 1016ms
Thread-1 Execution Time: 1016ms
Thread-21 Execution Time: 1015ms
Thread-25 Execution Time: 1014ms
Thread-29 Execution Time: 1014ms
Thread-4 Execution Time: 1014ms
Thread-6 Execution Time: 1014ms
Thread-2 Execution Time: 1015ms
Thread-14 Execution Time: 1007ms
Thread-22 Execution Time: 1014ms
Thread-20 Execution Time: 1014ms
Thread-0 Execution Time: 1014ms
Thread-12 Execution Time: 1014ms
Thread-19 Execution Time: 1015ms
Thread-28 Execution Time: 1014ms
Thread-9 Execution Time: 1014ms
Thread-8 Execution Time: 1016ms
Thread-31 Execution Time: 1015ms
Thread-3 Execution Time: 1014ms
Thread-11 Execution Time: 1015ms
Thread-26 Execution Time: 1015ms
Thread-17 Execution Time: 1015ms
Thread-24 Execution Time: 1014ms
Thread-10 Execution Time: 1007ms
Thread-15 Execution Time: 1014ms
Thread-18 Execution Time: 1014ms
Thread-7 Execution Time: 1014ms
Thread-5 Execution Time: 1016ms
Thread-13 Execution Time: 1014ms
Program Execution Time: 1077ms
1...200000 prime# counter=17984
```

How to compile and execute



```
C:₩Users₩a₩multicore₩proj1₩problem1>javac pc_static_block.java

C:₩Users₩a₩multicore₩proj1₩problem1>javac pc_static_cyclic.java

C:₩Users₩a₩multicore₩proj1₩problem1>javac pc_dynamic.java

C:₩Users₩a₩multicore₩proj1₩problem1>dir
 C 드라이브의 볼륨에는 이름이 없습니다.
 볼륨 일련 번호: 784D-CE34

 C:₩Users₩a₩multicore₩proj1₩problem1 디렉터리

2022-04-27  오후 07:38    <DIR>          .
2022-04-27  오후 07:38    <DIR>          ..
2022-04-28  오후 06:31             1,469 BlockThread.class
2022-04-28  오후 06:31             1,873 CyclicThread.class
2022-04-28  오후 06:32             2,432 DynamicThread.class
2022-04-28  오후 06:32             2,123 pc_dynamic.class
2022-04-27  오후 07:45             2,985 pc_dynamic.java
2022-04-07  오후 01:40             1,473 pc_serial.class
2022-04-23  오전 03:26               840 pc_serial.java
2022-04-28  오후 06:31             2,087 pc_static_block.class
2022-04-23  오전 03:26             2,300 pc_static_block.java
2022-04-28  오후 06:31             2,018 pc_static_cyclic.class
2022-04-23  오전 05:51             2,384 pc_static_cyclic.java
2022-04-23  오후 10:11           672,491 멀컴 레포트1.docx
              12개 파일             694,475 바이트
               2개 디렉터리  134,205,530,112 바이트 남음
```

Just use 'javac' to compile in my directory and run 'java pc_dynamic #num_thread #num_range' like below



```
C:₩Users₩a₩multicore₩proj1₩problem1>java pc_dynamic 8 200000
Thread-3 Execution Time: 1149ms
Thread-1 Execution Time: 1148ms
Thread-0 Execution Time: 1148ms
Thread-6 Execution Time: 1149ms
Thread-5 Execution Time: 1148ms
Thread-4 Execution Time: 1148ms
Thread-7 Execution Time: 1148ms
Thread-2 Execution Time: 1148ms
Program Execution Time: 1171ms
1...200000 prime# counter=17984
```