Objective(s):

You are to run benchmark on isPrime0, isPrime1, and isPrime2 mentioned in lectures.

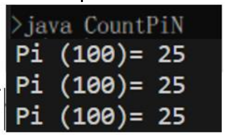
For computing numPrime(n), we can use the following program:

```
public class L2_CountPrimeNum {
  private static boolean isPrime0(int n) {
    if (n == 1) return false;
    if (n <= 3) return true;
    int m = n/2;
    for (int i = 2; i <= m; i++) {
      if (n % i == 0) return false;
    }
    return true;
  }
  private static void testIsPrime012() {
    int N = 100;
    int count = 0;
    for (int n = 1; n < N; n++) {
      if (isPrime0(n)) count++;
    }
    println("Pi ("+ N + ")= " + count);
    count = 0;
    for (int n = 1; n < N; n++) {
      if (isPrime1(n)) count++;
    }
    println("Pi ("+ N + ")= " + count);
    count = 0;
    for (int n = 1; n < N; n++) {
      if (isPrime2(n)) count++;
    }
    println("Pi ("+ N + ")= " + count);
  }
  public static void main(String [] args) {
    int count = 0;
    int N = 100;
    testIsPrime012();
  }
}
```

The method isPrime0(n) takes any positive integer and returns true if it is a prime, false otherwise. The method run through all integer from 2 to n/2 and check if n is divisible by any of them.

There are two more methods, isPrime1(n) and isPrime2(n). The method isPrime1(n) is similar to isPrime0(n) but only run from 2 to $\sqrt{n}$. The method isPrime2(n) improves upon isPrime1(n) by take out anything divisible by 2 and 3 and not going to test divisibility of number that are multiple of 2 and 3.

```
  private static boolean isPrime1(int n) {
    if (n == 1) return false;
    if (n <= 3) return true;
    int m = (int)Math.sqrt(n);
    for (int i = 2; i <= m; i++) {
      if (n % i == 0) return false;
    }
    return true;
  }
  private static boolean isPrime2(int n) {
    if (n == 1) return false;
    if (n <= 3) return true;
    if ((n%2 == 0) || (n%3 == 0)) return false;
    int m = (int)Math.sqrt(n);
    for (int i = 5; i <= m; i += 6) {
      if (n % i == 0) return false;
      if (n % (i+2) == 0) return false;
    }
    return true;
  }
```

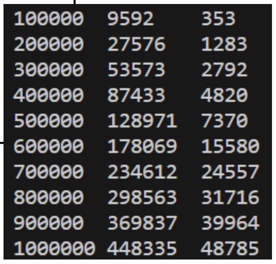


```
>java CountPiN
Pi (100)= 25
Pi (100)= 25
Pi (100)= 25
```

The output would look like this.

To measure efficiency of these methods, modify the main method as follows.

```
  public static void main(String [] args) {
    int count = 0;
    int N = 100;
    // testIsPrime012();
    for (N = 100_000; N <= 1_000_000; N+= 100_000) {
      long start = System.currentTimeMillis();
      for (int n = 1; n < N; n++) {
        if (isPrime0(n)) count++;
      }
      long time = (System.currentTimeMillis() - start);
      println(N + "\t" + count + "\t" + time);
    }
  }
}
```

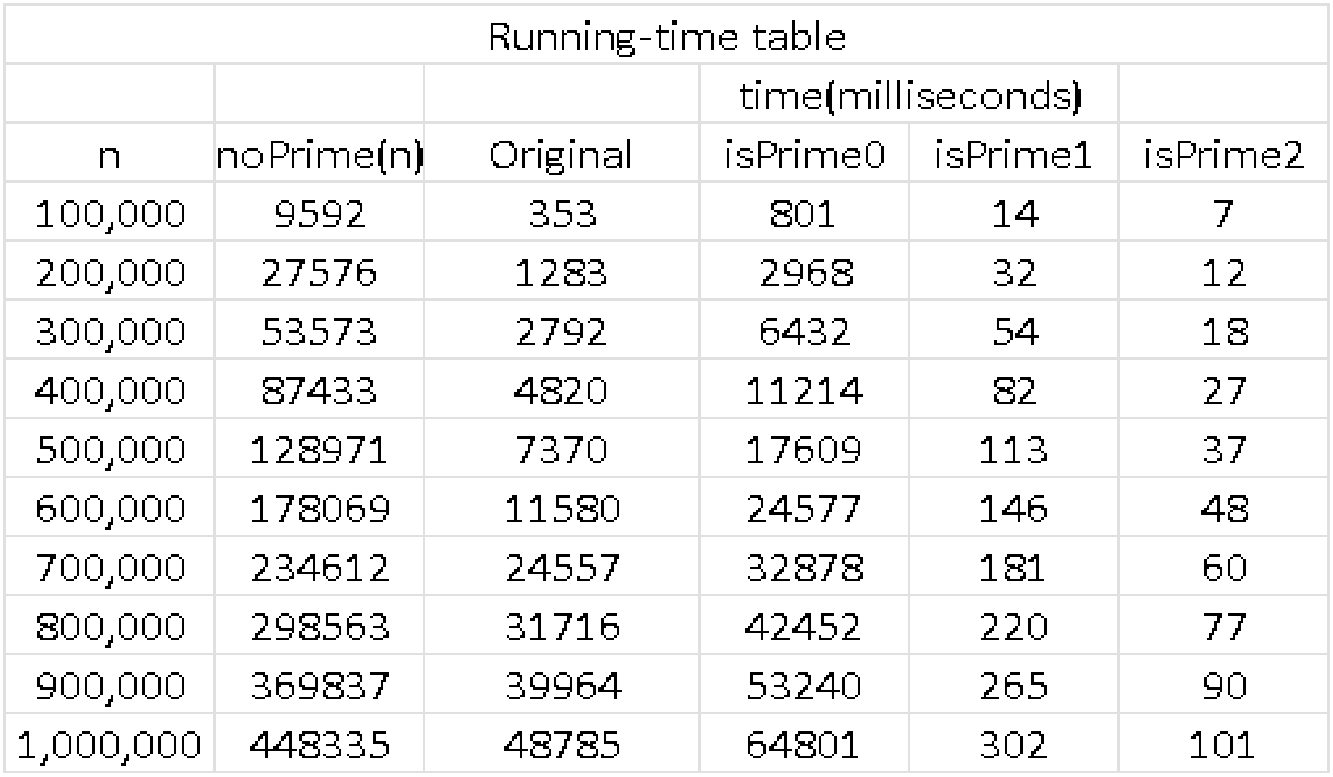


```
100000  9592    353
200000  27576   1283
300000  53573   2792
400000  87433   4820
500000  128971  7370
600000  178069  15580
700000  234612  24557
800000  298563  31716
900000  369837  39964
1000000 448335  48785
```

The result of running with isPrime0(n) should be

01286222 Lab 2 Name...Chanasorn Howattanakulphong..................... id ...65011277...................

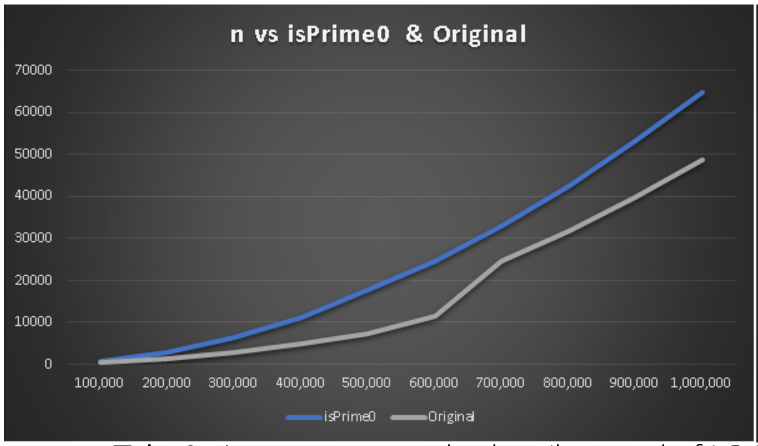
Task 1: run the program with isPrime0, isPrime1, and isPrime2. Record your result into the following table.

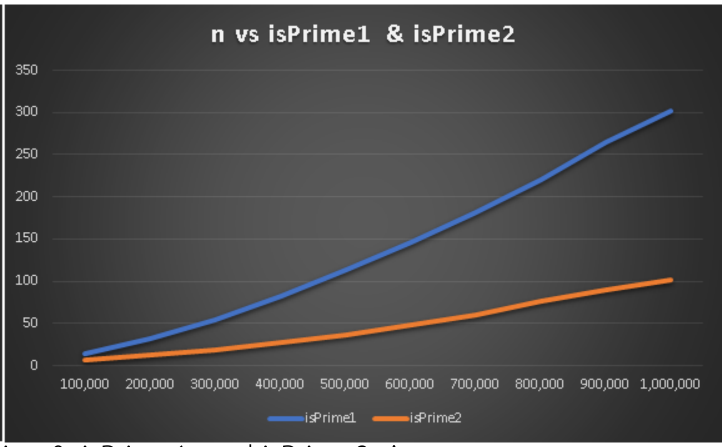


Running-time table

			time(milliseconds)		
n	noPrime(n)	Original	isPrime0	isPrime1	isPrime2
100,000	9592	353	801	14	7
200,000	27576	1283	2968	32	12
300,000	53573	2792	6432	54	18
400,000	87433	4820	11214	82	27
500,000	128971	7370	17609	113	37
600,000	178069	11580	24577	146	48
700,000	234612	24557	32878	181	60
800,000	298563	31716	42452	220	77
900,000	369837	39964	53240	265	90
1,000,000	448335	48785	64801	302	101

Task 2: Plot two graphs, A --> n vs. lab's result isPrime0's time and your isPrime0's and B --> n vs. lab's isPrime1's time and isPrime2's time. And your isPrime1's and isPrime2's





Taks 3 : In your own words, describe trend of isPrime0, isPrime1, and isPrime2. Are your recorded times differ to the recorded time given? Why?

The graphs isPrime0, isPrime1, isPrime2, Original are exponentially increasing as we increase the number of n. My recorded time was a bit slower than the original time, this might be due to the difference in cpu's performance.

Due Date: TBA