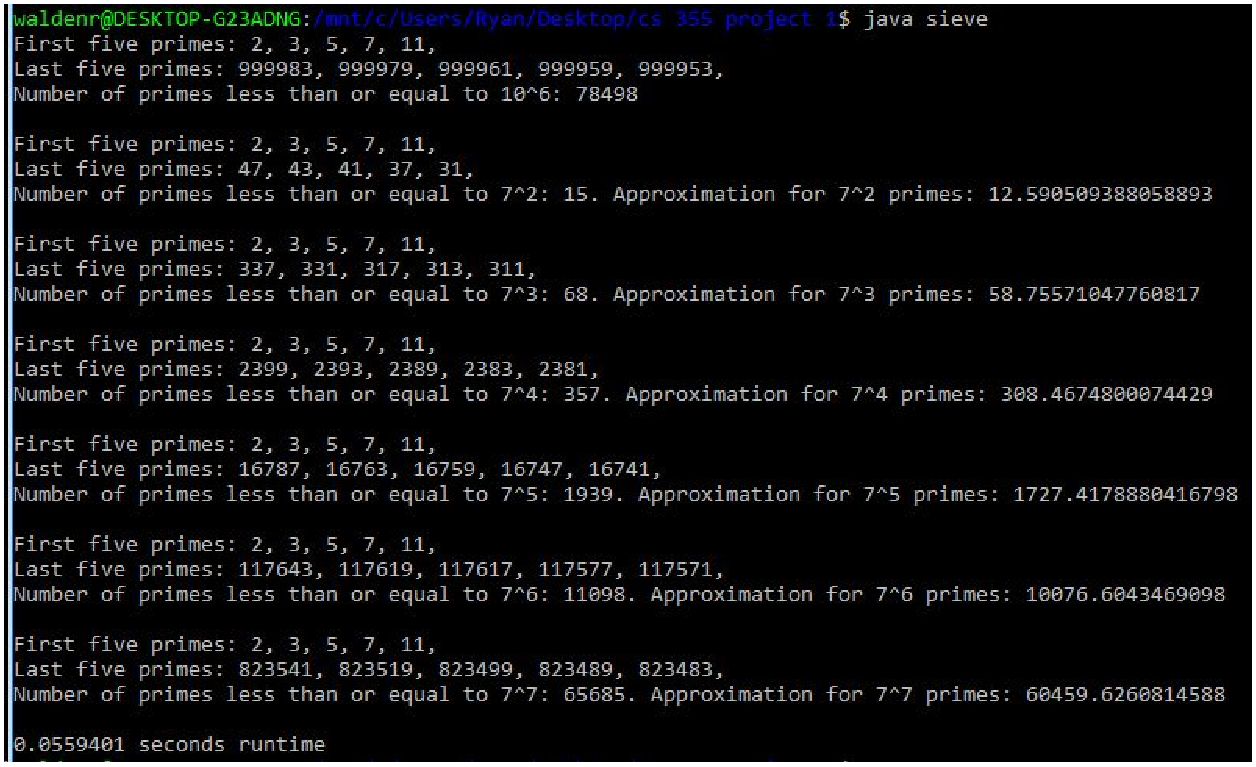
CS 355 Project 1

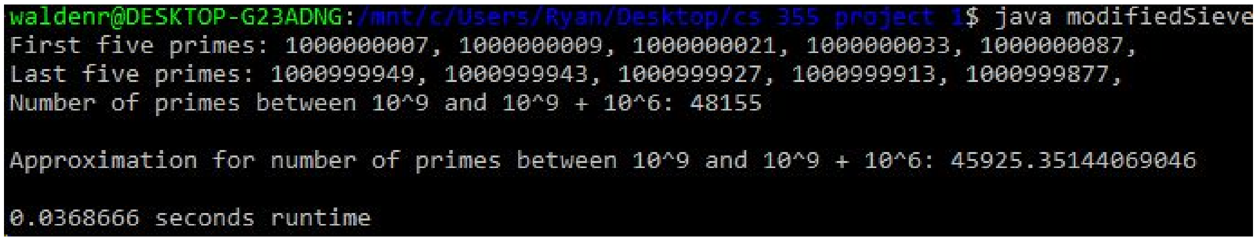
Ryan Walden

Part A:



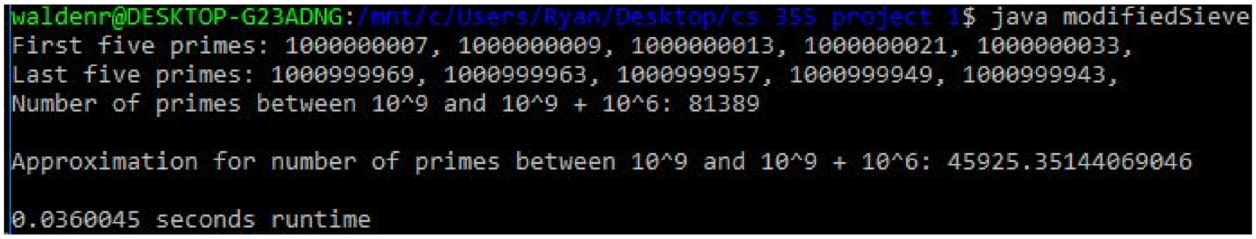
```
waldenr@DESKTOP-G23ADNG:/mnt/c/Users/Ryan/Desktop/cs 355 project 1$ java sieve
First five primes: 2, 3, 5, 7, 11,
Last five primes: 999983, 999979, 999961, 999959, 999953,
Number of primes less than or equal to 10^6: 78498

First five primes: 2, 3, 5, 7, 11,
Last five primes: 47, 43, 41, 37, 31,
Number of primes less than or equal to 7^2: 15. Approximation for 7^2 primes: 12.590509388058893

First five primes: 2, 3, 5, 7, 11,
Last five primes: 337, 331, 317, 313, 311,
Number of primes less than or equal to 7^3: 68. Approximation for 7^3 primes: 58.75571047760817

First five primes: 2, 3, 5, 7, 11,
Last five primes: 2399, 2393, 2389, 2383, 2381,
Number of primes less than or equal to 7^4: 357. Approximation for 7^4 primes: 308.4674800074429

First five primes: 2, 3, 5, 7, 11,
Last five primes: 16787, 16763, 16759, 16747, 16741,
Number of primes less than or equal to 7^5: 1939. Approximation for 7^5 primes: 1727.4178880416798

First five primes: 2, 3, 5, 7, 11,
Last five primes: 117643, 117619, 117617, 117577, 117571,
Number of primes less than or equal to 7^6: 11098. Approximation for 7^6 primes: 10076.6043469098

First five primes: 2, 3, 5, 7, 11,
Last five primes: 823541, 823519, 823499, 823489, 823483,
Number of primes less than or equal to 7^7: 65685. Approximation for 7^7 primes: 60459.6260814588

0.0559401 seconds runtime
```

Part B:



```
waldenr@DESKTOP-G23ADNG:/mnt/c/Users/Ryan/Desktop/cs 355 project 1$ java modifiedSieve
First five primes: 1000000007, 1000000009, 1000000021, 1000000033, 1000000087,
Last five primes: 1000999949, 1000999943, 1000999927, 1000999913, 1000999877,
Number of primes between 10^9 and 10^9 + 10^6: 48155

Approximation for number of primes between 10^9 and 10^9 + 10^6: 45925.35144069046

0.0368666 seconds runtime
```

Part C:



```
waldenr@DESKTOP-G23ADNG:/mnt/c/Users/Ryan/Desktop/cs 355 project 1$ java modifiedSieve
First five primes: 1000000007, 1000000009, 1000000013, 1000000021, 1000000033,
Last five primes: 1000999969, 1000999963, 1000999957, 1000999949, 1000999943,
Number of primes between 10^9 and 10^9 + 10^6: 81389

Approximation for number of primes between 10^9 and 10^9 + 10^6: 45925.35144069046

0.0360045 seconds runtime
```

The primes between I and J are crossed out, because this is the nature of the algorithm, we assume all numbers are prime then cross them out as we iterate through the range and find composites. Some of the composites are not crossed out, even though they should be. This is because 1000 < sqrt(J) so the composite checker does not check every number between I and J.

Part A code:

```
import java.lang.Math;
public class sieve {

        public static int sieve(int J){

        if(J <= 2){
        return 0;
        }
        if(J == 3){
        return 1;
        }
        int count = 0;
        boolean[] P = new boolean[J+1];
        P[J] = true;
        for (int i = 2; i*i <= J; i++) {
        if(!P[i]){
                for (int j = i*i; j <= J; j+=i) {
                P[j] = true;
                }
        }
        }
        int firstFive = 5;
        int lastFive = 5;
        System.out.print("First five primes: ");
        for (int i = 2; i <= J; i++) {
        if(!P[i]){
                count++;
                if (firstFive > 0) {
                System.out.print(i + ", ");
                firstFive--;
                }
        }
        }
        System.out.print("\nLast five primes: ");
        for (int i = J; i >= 0; i--) {
        if(!P[i]){
        if (lastFive > 0) {
                System.out.print(i + ", ");
                lastFive--;
        }
        }
        }
        return count;
        }

        public static void main(String[] args) {
        long startTime = System.nanoTime();
        System.out.println("\nNumber of primes less than or equal to 10^6: " + sieve(1000000) +
"\n");
        System.out.println("\nNumber of primes less than or equal to 7^2: " + sieve(49) + ".
Approximation for 7^2 primes: " + (49) / (Math.log(49)) + "\n");
        System.out.println("\nNumber of primes less than or equal to 7^3: " + sieve(343) + ".
Approximation for 7^3 primes: " + (343) / (Math.log(343)) + "\n");
        System.out.println("\nNumber of primes less than or equal to 7^4: " + sieve(2401) + ".
Approximation for 7^4 primes: " + (2401) / (Math.log(2401)) + "\n");
        System.out.println("\nNumber of primes less than or equal to 7^5: " + sieve(16807) + ".
Approximation for 7^5 primes: " + (16807) / (Math.log(16807)) + "\n");
        System.out.println("\nNumber of primes less than or equal to 7^6: " + sieve(117649) + ".
Approximation for 7^6 primes: " + (117649) / (Math.log(117649)) + "\n");
        System.out.println("\nNumber of primes less than or equal to 7^7: " + sieve(823543) + ".
Approximation for 7^7 primes: " + (823543) / (Math.log(823543)) + "\n");
        long endTime   = System.nanoTime();
        System.out.println((endTime - startTime)/1000000000.0 + " seconds runtime");
        }
}
```

Part B code:

```
import java.lang.Math;
```

```
public class modifiedSieve {

        public static int modifiedSieve(int I, int J){

        int L = (int)Math.sqrt(J); //for Part B
        //int L = 1000; for Part C
        int count = 0;
        boolean[] P = new boolean[L+1];
        P[1] = true;
        for (int i = 2; i*i <= L; i++) {
        if(!P[i]){
                for (int j = i*i; j <= L; j+=i) {

                P[j] = true;
                }
        }
        }
        int[] A = new int[1000000 + 1];
        for(int i = 0; i < 1000000; i++) {
        A[i]=1;
        }
        A[0] = 0;
        int p;
        for (p=1; p <= L; p++) {
        if (!P[p]) {
        int i = I/p;
        i = (i + 1) * p;
        //  System.out.println(p);
        //System.out.println(J);
        if (p == 1)
        System.out.println(i);
        while (i < J) {
        //System.out.println("stuck there");
        //System.out.println(i);
        A[i - I] = 0;
        i = i + p;
        }
        }
        }
        int firstFive = 5;
        int lastFive = 5;
        System.out.print("First five primes: ");
        for (int i = I; i < J; i++) {
        if(A[i - I] == 1){
        count++;
        if (firstFive > 0) {
                System.out.print(i + ", ");
                firstFive--;
        }
        }
        }
        System.out.print("\nLast five primes: ");
        for (int i = J; i >= I; i--) {
        if(A[i - I] == 1){
        if (lastFive > 0) {
                System.out.print(i + ", ");
                lastFive--;
        }
        }
        }

        return count;
        }

        public static void main(String[] args) {
        //System.out.println(((int)Math.pow(10, 6) + (int)Math.pow(10, 8)));
        long startTime = System.nanoTime();
        System.out.println("\nNumber of primes between 10^9 and 10^9 + 10^6: " +
modifiedSieve(1000000000, 1001000000 ) + "\n");
        System.out.println("Approximation for number of primes between 10^9 and 10^9 +
10^6: " + (1001000000/(Math.log(1001000000)) - (1000000000)/(Math.log(1000000000))) +
"\n");
        long endTime   = System.nanoTime();
        System.out.println((endTime - startTime)/1000000000.0 + " seconds runtime");
        }
}
```