// Print out the prime numbers that are less than the given number.

According to Vedic mathematics (http://vedicriver.blogspot.ca/), it says if a number n is prime then it must satisfy this formula:

((2^(n-1)) mod n) - 1 = 0

Testing of the formula:

For n = 5, 2^4 mod 5 =16 mod 5 =1. The formula is satified.

For n = 13, 2^12 mod 13=4096 mod 13 = 1. The formula is also satisfied.

For n = 6, 2^5 mod 6 =32 mod 6 =2 $\neq$1. The formual is not satifisfied.

Similarly we can check for all other prime numbers.

Formula Time Complexity:

Since power and modulo operator takes constant time, the formula takes constant time to testify whether the number is prime or not. Therefore, its time complexity: O(1).

Algorithm Time Complexity:

O(N)

Java version

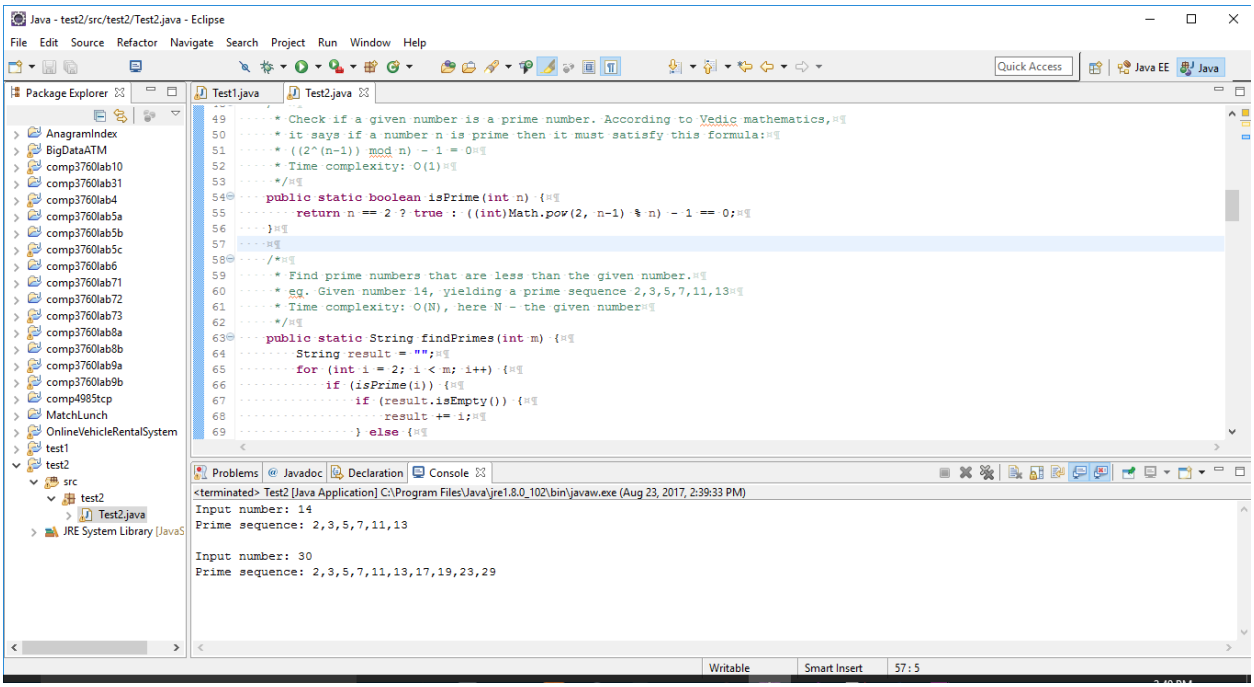
```
/*
 * Check if a given number is a prime number. According to Vedic mathematics,
 * it says if a number n is prime then it must satisfy this formula:
 * ((2^(n-1)) mod n) - 1 = 0
 * Time complexity: O(1)
 */
public boolean isPrime(int n) {
    return n == 2 ? true : ((int)Math.pow(2, n-1) % n) - 1 == 0;
}

/*
 * Find prime numbers that are less than the given number.
 * eg. Given number 14, yielding a prime sequence 2,3,5,7,11,13
 * Time complexity: O(N), here N - the given number
 */
public String findPrimes(int m) {
    String result = "";
    for (int i = 2; i < m; i++) {
        if (isPrime(i)) {
```

```
                if (result.isEmpty()) {
                    result += i;
                } else {
                    result += "," + i;
                }
            }
        }

        return result;
}
```

Output screenshot:



C# version

```
/*
 * Check if a given number is a prime number. According to Vedic mathematics,
 * it says if a number n is prime then it must satisfy this formula:
 * ((2^(n-1)) mod n) - 1 = 0
 * Time complexity: O(1)
 */
public bool isPrime(int n)
{
    return n == 2 ? true : ((int)Math.Pow(2, n - 1) % n) - 1 == 0;
}

/*
 * Find prime numbers that are less than the given number.
 * eg. Given number 14, yielding a prime sequence 2,3,5,7,11,13
 * Time complexity: O(N), here N - the given number
```

```
 */
public String findPrimes(int m)
{
    String result = String.Empty;
    for (int i = 2; i < m; i++)
    {
        if (isPrime(i))
        {
            if (String.IsNullOrEmpty(result))
            {
                result += i;
            }
            else {
                result += "," + i;
            }
         }
     }

     return result;
}
```

Output screenshot: