**Lab 1 // The Sieve of Eratosthenes and the Goldbach Conjecture**

**CS210 – Algorithms and Data Structure (Dr. Pierpaolo Dondio)**

---

## 1. Objectives

By the end of this lab, you should be able to:

1. Load and use the **Sieve of Eratosthenes** to generate prime numbers.
2. Compute the number of comparisons that the algorithm performs when the range of numbers increases.
3. Implement a function to verify the **Goldbach Conjecture for a given number**.

Part 1.
In class we presented a Java implementation of the Sieve of Eratosthenes. Download the file **sieve.java** from Moodle Week 1 and run it.
The variable **limit** sets the interval of numbers in which we want to identify prime numbers. For instance, if limit = 100, the algorithm will find all the prime numbers up to 100. Using the algorithm, find the number of prime numbers up to 1000.

Part 2.
We will now study how "complex" the algorithm is by counting how many basic operations it needs to perform in order to find its solution when we increase the range of numbers.
Declare a new variable **counter** inside the sieve class. Declare it as a public static variable. This allows the variable to be visible in all parts of your program (similar to a global variable). Declare it like this:

```
public class sieve {

    public static int counter=0;

     …
```

Then, add the following instruction:

```
counter++;
```

inside the innermost loop of the Sieve of Eratosthenes function. The innermost loop of the function is:

```
for (int i = p * p; i <= n; i += p) {

    prime[i] = false;

    }
```

Now run the program and print the value of the **counter** variable. When limit = 100, **counter** should have a value of 104.

Run the programs multiple times with different values of the limit variable and fill the following table. You can run the program 20 times or write a loop to compute the output in one run.

| Value of the limit variable | Value of the counter variable (=number of steps performed by the algorithm) |
|---|---|
| 50 | |
| 100 | |
| 150 | |
| 200 | |
| 250 | |
| 300 | |
| 350 | |
| 400 | |
| 450 | |
| 500 | |
| 550 | |
| 600 | |
| 650 | |
| 700 | |
| 750 | |
| 800 | |
| 850 | |
| 900 | |
| 950 | |
| 1000 | |

Draw a graph of your results, where the x-axis represents the variable **limit** and the y-axis represents the variable **counter**. Can you describe the shape of the graph? Does it look like a line, or is it a curve? If it is a curve, does it go up faster than a line, or slower?

**Part 3. The Goldback Conjecture**

Proposed in 1742 by Christian Goldbach, the conjecture states:

*Every even integer greater than 2 can be expressed as the sum of two prime numbers.*

Some numbers can have multiple ways of being expressed as the sum of two prime numbers, but they have at least one.

This conjecture has been verified up to very large numbers, but never proven theoretically, and it is still one of the greatest unsolved maths theorems.

Examples:

- 8 = 3 + 5

- 20 = 3 + 17 = 7 + 13  (two possible sums)
- 42 = 5 + 37 = 11 + 31 = 19 + 23  (three possible sums)

Using the Sieve of Eratosthenes, you are required to create a Java program to verify, for each positive even number, if the Goldbach Conjecture is true. Print out the output.

For instance, if the input is n = 14, your program should output:

14 = 3 + 11  (or 14=7+7), one solution is enough.

**HINT:** Given the input number n (positive and even), the Sieve of Eratosthenes can help you find all the prime numbers up to n. Then, using this information, you must find two of these prime numbers that add up to n.