**Poulet Numbers:**

A Pouletnumber or Sarrusnumber is an odd composite number n such that:

https://number.subwiki.org/w/images/math/6/c/c/6ccfccf4fed28875f647d108dc8ddac2.png.

In other words, https://number.subwiki.org/w/images/math/7/b/8/7b8b965ad4bca0e41ab51de7b31363a1.png divides https://number.subwiki.org/w/images/math/6/1/d/61d44b0f82ed39d9edb64a420b6477f4.png.

Equivalently, n is a Fermat Pseudoprime in base 2.

An odd composite number is the product of two whole numbers greater than one*.*

The first few Poulet numbers are:

341, 561, 645, 1105, 1387, 1729, 1905, 2047, 2465, 2701, 2821, 3277, 4033, 4369, 4371, 4681, 5461, 6601, 7957

**Solution Approach:**

Created a function that verifies if a number is composite.

The program in itself, will take the number of thread it is running on (starting from 1) and it will multiply it with 2 and add 1, this way the first number it starts with is 3. This number called ‘p’ will be incremented by 2 times the number of threads, so for example if we had 1 thread it would increase by 2, from 3 to 5, this way the number remains odd (it helps performance that we verify only odd numbers if they are composite).

At each iteration this number ‘p’ is verified if it is composite. If it happens to be composite, we verify the condition that it is a Poulet number if this number minus one as exponent of 2 will gives us a remainder of 1 if divided by itself.

If it meets this requirement as well, it is a Poulet number and it will be printed.

My CPU (Intel i7-4720HQ) has 4 cores.

For each language, I used the same approach, only the syntax has changed.

For the C with OpenMP and with Pthreads, I used the GMP library for making calculations with big numbers and n Java, I used BigDecimal.