Programme en Perl de Damian Conway pour calculer les décompositions de Goldbach de nombres pairs (juin 2013), Denise Vella-Chemla, juillet 2026

On peut tester ce programme en le copiant sur une plateforme d'exécution de Perl en ligne, par exemple à cette adresse https://www.jdoodle.com/execute-perl-online. Le programme en ligne ne réussit pas à calculer les décompositions de 10^6 (allowed output limit reached).

```
NUMBER:
for my $n (1..100, 200, 500, 1000, 1e4, 1e5, 1e6) {
    # Skip odd numbers...
    next NUMBER if $n %2 != 0;
    # Locate components...
    my @GC = find_GC($n);
    # Print table row...
    print $n;
    for my $gc (@GC) {
        print ' = ',$gc,' + ', $n-$gc;
    }
    print "\n"
}

# Return list of Goldbach components...
sub find_GC {
    my($n) = @_ ;
    # All m_i that are prime and
    # > my @m_i = grep {is_prime($_)} 2..sqrt($n);
    # Potential values of p...
    my @p = 2..$n/2;
    # Return values of p meeting GC criterion...
    return grep {my $p=$_; is_prime($p) && not_equiv($p, $n, @m_i)}@p;
}

# Modular non-equivalence...
sub not_equiv {
    my ($x, $y, @mod_set) = @_;
    # If there exists a modulus in the set...
    for my $modulus (@mod_set) {
        # ...such that x and y are equal (modulo that modulus),
        # then return false, because they are equivalent...
        return 0 if $x % $modulus == $y % $modulus;
    }
    # If no such modulus, then they are not equivalent, so return true...
    return 1;
}

# Simple primality test with caching of previous results...
my %known_prime;
sub is_prime {
    my ($n) = @_;
    # Check cache first...
    return 1 if $known_prime{$n};
    # Search (inefficiently)...
```

```
    for my $divisor (2..sqrt($n)+1) {
        # Not a prime, if divisible by anything...
        return 0 if $n % $divisor == 0;
    }
    # Cache new prime...
    return $known_prime{$n} = 1;
    # Return true, indicating it is a prime...
    return 1;
}
```