Проверка на простоту некоторых чисел вида $2 \cdot 3^m-1$

№	m	Количество десятичных разрядов	Время проверки, целая часть секунд
1	1	1	0
2	2	2	0
3	7	4	0
4	8	5	0
5	12	7	0
6	23	12	0
7	56	28	0
8	62	30	0
9	68	33	0
10	131	63	0
11	387	185	0
12	644	308	0
13	3751	1790	1
14	5270	2515	2
15	6335	3023	3
16	8544	4077	6
17	9204	4392	7

В программе реализован алгоритм, описанный Е.В. Садовником в статье «Проверка на простоту некоторых чисел вида $2kp^m-1$» при k=1, p=3. Программа реализована на языке C++ с использованием библиотеки GMP (https://gmplib.org/) и запускалась на компьютере со следующими характеристиками:

процессор - Pentium(R) Dual-Core CPU T4400 @ 2.20GHz × 2;
память - 3,9 ГиБ.

За 24 часа было проверено 14000 чисел указанного вида, среди них найдено 16 простых (см. таблицу).
Текст программы с комментариями:

```
#include <gmp.h>
#include <gmpxx.h>
#include <time.h>
#include <iostream>
#include <fstream>

using namespace std;

int main (void)
{
   mpz_t n, s, k, s_tmp, tmp;
```

```
        unsigned int m, j;
        time_t t1, t2;
        mpz_inits(n, s, s_tmp, k, tmp, NULL);
        mpz_set_ui(n, 5);

        ofstream prime, composite;
        prime.open("prime.txt", fstream::trunc);
        composite.open("composite.txt", fstream::trunc);

        for (m = 2; m <= 49340; ++m)
    {
      /* n = n * 3 + 2; */
      mpz_mul_ui(n, n, 3);
                mpz_add_ui(n, n, 2);

            t1 = time(NULL);
            mpz_set_ui(s, 14);
            for (j = 2; j <= m; ++j) {

                /* s = (s * (s * s - 3)) % n; */
                mpz_set(s_tmp, s);
                mpz_mul(s, s, s);
                mpz_sub_ui(s, s, 3);
                mpz_mul(s, s, s_tmp);
        mpz_mod(s, s, n);
      }
      t2 = time(NULL);

      /* if (s != 2) { */
                if (mpz_cmp_ui(s, 2) != 0) {

                /* k = s * s - 1; */
                mpz_mul(k, s, s);
                mpz_sub_ui(k, k, 1);

                /* if (!(k % n)) */
                mpz_mod(tmp, k, n);
                if (mpz_cmp_ui(tmp, 0) == 0)
                  /*true*/
                  prime << m << endl << mpz_get_str(NULL, 10, n) << " " << (t2 - t1) << endl;
                else
                  /*false*/
                  composite << m << endl;
                }
        }

        mpz_clears(n, s, s_tmp, k, tmp, NULL);
        prime.close();
        composite.close();

    return 0;
}
```