```
//Программа написана на языке C++ с использованием библиотеки
для работы с большими числами gmp
//И реализует алгоритм для проверки простых чисел вида 2*k*3^m
- 1, описанный в статье Е. В. Садовник “Проверка на простоту
некоторых чисел вида N=2kp^m-1”, Дискрет. матем., 18:1 (2006),
146–155
http://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=dm&p
aperid=38&option_lang=rus
// Программа запускалась под операционную систему Mac OS X 10


#include <stdlib.h>
#include <stdio.h>
#include <stdint.h>
#include <sys/time.h>
#include <inttypes.h>
#include <gmp.h>
#include <math.h>
//#include <iostream>

/* #define LIM 55000 */
#define LIM 100

int
main(int argc,char *argv[])
{

   // FILE *fil;
    mpz_t n, s, k, s_pres, l, l_tmp, tmp, l_tmp_tmp, new_num,
l_criteria;
    mpz_t cur_n;
    mpz_t s1, s2, s1_next, s2_next;
    char p = ' ';
    //char *p11 = (char*)argv[1];//"out2.txt";

    unsigned int j;
    mpz_t par;
    int cur_pow = 1;
    //unsigned long cur_pow = 1;
    clock_t time;

    signed int cur_dig;

   // fprintf(stdout,"%s", argv[1]);
    //return 0;

    mpz_inits(n, s, s1, s2, s1_next, s2_next, s_pres, k, l,
l_tmp,l_tmp_tmp, l_criteria, par, tmp, new_num, cur_n, NULL);
    mpz_set_ui(cur_n, 3);
    mpz_set_ui(l, 1);


    //cur_n - 3^n
    //n - number that we check without koef
```

```
    //l-koefficient
    //new_num - number with koef

    //mpz_set_ui(l, 173184579);
    //cur_pow = 2;//2010;
    //fprintf(stdout, "%lu", cur_pow);
    cur_pow = atoi(argv[1]);

    //cur_pow = 3;
    for(int i = 1; i < cur_pow; i++){
        mpz_mul_ui(cur_n, cur_n, 3);
    }

    do {
        mpz_mul_ui(cur_n, cur_n, 3);
        mpz_mul_ui(n, cur_n, 2);
        mpz_sub_ui(n, n, 1);
        cur_pow++;


        /*mpz_pow_ui(n, n, 20); ///TODO change power!!! to m
        mpz_mul_ui(n, n, 2);
        mpz_sub_ui(n, n, 1);
        gettimeofday(&t2, NULL);
        printf("took %lu\n", (unsigned long)(t2.tv_usec -
t1.tv_usec));*/


        mpz_set_ui(l, 1);
        mpz_set_ui(l_criteria, 1);
        while(mpz_cmp(l_criteria, cur_n) < 0){
            mpz_mul_ui(l_criteria, l, 2);

            mpz_set_si(s1_next, -14);
            mpz_set_ui(s2_next, 1);
            //fprintf(stdout, "Calculating power = %i and k
=%s \n", cur_pow, mpz_get_str(NULL, 10, l));
            time = clock();



            mpz_add_ui(new_num, n, 1);
            mpz_mul(new_num, new_num, l);
            mpz_sub_ui(new_num, new_num, 1);

            mpz_set(l_tmp_tmp, l);
            mpz_set_ui(l_tmp, 1);
            while(mpz_cmp_ui(l_tmp_tmp,1) > 0){
                mpz_mul_ui(l_tmp, l_tmp, 2);
                mpz_mod_ui(par, l_tmp_tmp, 2);
                mpz_add(l_tmp, l_tmp, par);
                mpz_div_ui(l_tmp_tmp, l_tmp_tmp, 2);
            }
```

```
            while(mpz_cmp_ui(l_tmp, 1) != 0){
                mpz_set(s1, s1_next);
                mpz_set(s2, s2_next);
                mpz_mod_ui(tmp, l_tmp, 2);
                cur_dig = (unsigned int)mpz_get_ui(tmp) * 8;
                //s1_next calculate
                mpz_mul(tmp, s1, s1);
                mpz_sub_ui(tmp, tmp, 2);
                mpz_set(s1_next, tmp);
                mpz_set(s2_next, tmp);
                mpz_mul_ui(tmp, tmp, cur_dig);
                mpz_sub(s1_next, s1_next, tmp);
                mpz_set_ui(tmp, cur_dig);
                mpz_mul_ui(tmp, tmp, 12);
                mpz_mul(tmp, tmp, s1);
                mpz_mul(tmp, tmp, s2);
                mpz_add(s1_next, s1_next, tmp);
                mpz_mod(s1_next, s1_next, new_num);
                //s1_next = (s1 * s1 - 2) * (1 - cur_dig) + 12
* cur_dig * s1 * s2;
                //s2_next calculate
               // mpz_mul(s2_next, s1, s1);
               // mpz_sub_ui(s2_next, s2_next, 2);
                mpz_mul_si(tmp, s1, cur_dig - 1);
                mpz_mul(tmp, tmp, s2);
                cur_dig /= 8;
                mpz_mul_ui(s2_next, s2_next, cur_dig);
                mpz_div_ui(s2_next, s2_next, 2);
                mpz_sub(s2_next, s2_next, tmp);
                mpz_mod(s2_next,s2_next, new_num);


                //s2_next = cur_dig * ((s1 * s1 - 2) - (-1 +
8cur_dig) * s1 * s2) / 2;

                //m--;



                mpz_div_ui(l_tmp, l_tmp, 2);
            }

          //  printf("Info: num =  %s\n\n", mpz_get_str(NULL,
10, new_num));

            mpz_set(s, s1_next);
            for (j = 1; j < cur_pow; j++) {
                mpz_set(s_pres, s);   //s_pres = s
                mpz_mul(s, s, s);     //s *= s
                mpz_sub_ui(s, s, 3);  //s -= 3
                mpz_mul(s, s, s_pres);//s *= s_pres
                mpz_mod(s, s, new_num);    //s = s % n
```

```
            }

            if (mpz_cmp_si(s, -2) != 0) {
                mpz_mul(k, s, s);     //k = s*s
                mpz_sub_ui(k, k, 1);  //k --;
                mpz_mod(tmp, k, new_num);   //tmp = k % n
                if (mpz_cmp_ui(tmp, 0) == 0){
                    //printf("new_num = %s time %f \n",
mpz_get_str(NULL, 10, new_num), (double)(clock() -
time)/CLOCKS_PER_SEC);
                    //fil = fopen("out2.txt", "r+");
                    //fseek(fil, 0, 2);
                    fprintf(stdout,
"num=%s\npow=%i\nkoef=%s\ntime=%f\n\n", mpz_get_str(NULL, 10,
new_num), cur_pow, mpz_get_str(NULL, 10, l), (double)(clock()
- time)/CLOCKS_PER_SEC);
                    //std::cout<<"num = "<<mpz_get_str(NULL,
10, new_num)<<"\nkoef="<<mpz_get_str(NULL, 10,
l)<<"\ntime="<<(clock() - time)/CLOCKS_PER_SEC << "\npow =
"<<cur_pow;
                    //fclose(fil);
                    //printf("\nTrue m = %u\n", m);
                }
            } else
                printf("Alarm! m = %i\n", cur_pow);

            mpz_add_ui(l, l, 2);
            mpz_mod_ui(l_tmp, l, 3);
            if(mpz_cmp_ui(l_tmp, 0) == 0){
                mpz_add_ui(l,l,2);
            }
            //printf("time %f \n", (double)(clock() -
time)/CLOCKS_PER_SEC);
        }
       // p = getch();
    }
    while(p != 27);
    mpz_clears(n, s, s_pres, k, tmp, l, new_num, NULL);

    return 0;
}
```