horizontal line

**Assignment 03 | Advance Algorithms**

**CE-092**

Assignment submission for Advance Algorithms subject week 3.

nevilparmar24@gmail.com

**─**

**Task 1:**

To Implement fermat’s primality testing algorithm.

Code:

/\*

*\* @Author: nevil*

*\* @Date: 2020-07-24 16:03:04*

*\* @Last Modified by: nevil*

*\* @Last Modified time: 2020-07-24 16:08:26*

\*/

#*include*<bits/stdc++.h>

using namespace std;

void *swap*(long long *&*a, long long *&*b) {

long long temp = a;

a = b;

b = temp;

}

long long *gcd*(long long a, long long m) {

*if* (a > m) *swap*(a, m);

long long r;

*for* (;;) {

r = m % a;

*if* (r == 0) *return* a;

m = a;

a = r;

};

}

long long *power*(long long a, long long b, long long m) {

*if* (a > m) *swap*(a, m);

long long c = 1;

*for* (;;) {

*if* (b % 2 == 1) c = (c \* a) % m;

b = b / 2;

*if* (b == 0) *return* c;

a = (a \* a) % m;

};

}

bool *fermat*(long long m) {

*srand*(*time*(*NULL*));

*if* (m == 1) {

cout *<<* *endl*;

*return* false;

};

*for* (int i = 0; i < 1000; i++) {

int cnt = i + 1;

long long a = *rand*() % m;

*if* (a == 0) a = 1;

*if* (*gcd*(a, m) != 1) {

cout *<<* "\nFailed at the " *<<* "attempt " *<<* cnt *<<* *endl*;

cout *<<* "Found a divisor: " *<<* *gcd*(a, m) *<<* ", thus ";

*return* false;

};

*if* (*power*(a, m - 1, m) != 1) {

cout *<<* "\nFailed at the " *<<* cnt *<<* ". attempt. ";

cout *<<* "Found a Fermat witness: " *<<* a *<<* ", thus ";

*return* false;

};

};

cout *<<* *endl*;

*return* true;

}

int *main*() {

cout *<<* "Fermat primality test" *<<* *endl* *<<* *endl*;

*for* (;;) {

long long n;

cout *<<* "Please input a natural number (0 to quit): ";

cin *>>* n;

*if* (n == 0) *return* 0;

bool b = *fermat*(n);

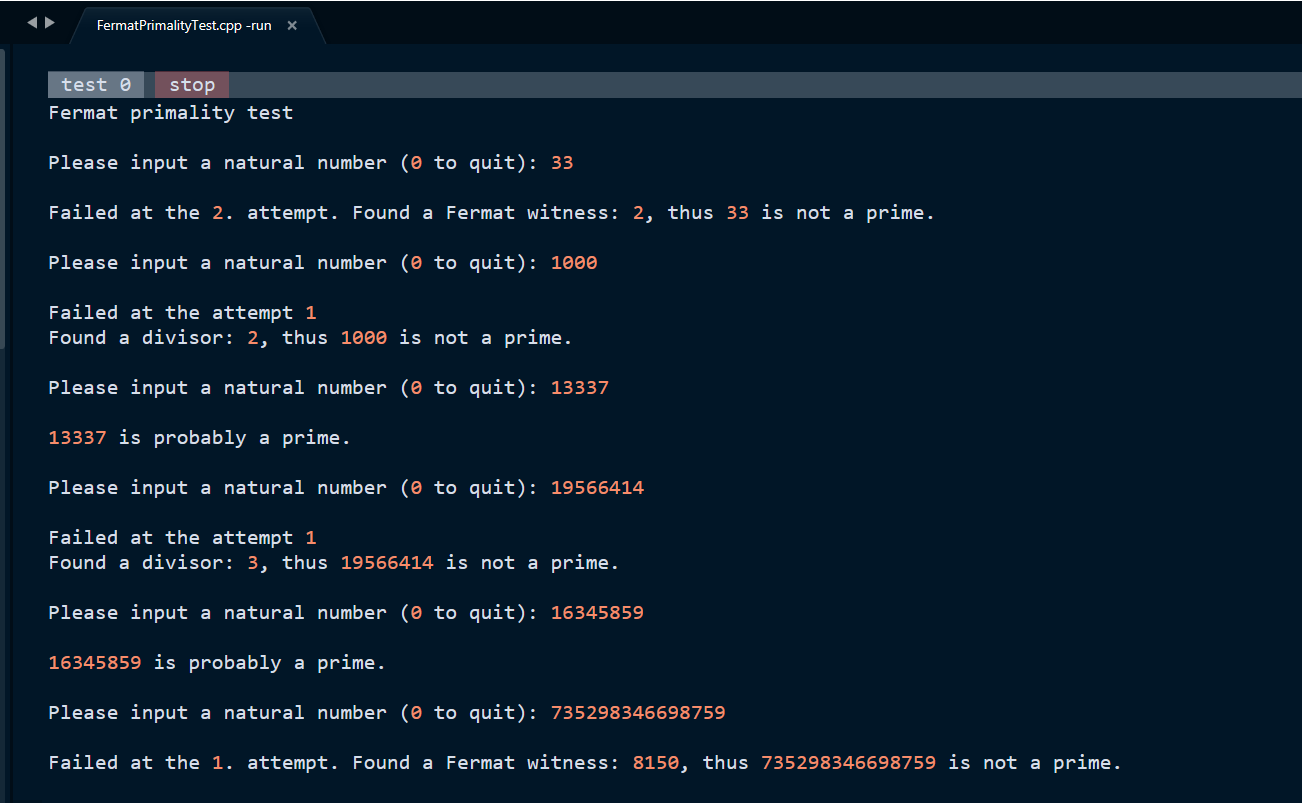
*if* (b) cout *<<* n *<<* " is probably a prime." *<<* *endl* *<<* *endl*;

*if* (!b) cout *<<* n *<<* " is not a prime." *<<* *endl* *<<* *endl*;

};

}

Output:



**Conclusion :**

If a given number is prime, then this method always returns true. If a given number is composite (or non-prime), then it may return true or false, but the probability of producing incorrect results for composite is low and can be reduced by doing more iterations.

**Complexity :**

The fermat’s theorem may fail even if we increase the number of iterations (higher k). There exist some composite numbers with the property that for every a < n, gcd(a, n) = 1 and a^n-1 ≡ 1 (mod n). Such numbers are called Carmichael numbers.

Considering the power method takes O(Logn) time. We can clearly say that the complexity of fermat’s primality testing algorithm is O(k \* Logn).

**Application of Fermat’s Test :**

Fermat’s primality test is often used if a rapid method is needed for filtering, for example in the key generation phase of the RSA public key cryptographic algorithm.

Nevil Parmar

CE-092

https://nevilparmar.me