

AA_LAB_03_Assignment

Aim :- Write a Program to Implement Randomized Primality Testing using Fermat's Method.

Code :-

```
# -*- coding: utf-8 -*-
```

```
"""
```

```
Created on Fri Jul 31 16:14:36 2020
```

```
@author: DHRUV
```

```
"""
```

```
import random as rn
```

```
def find_gcd(num1, num2):          # Find GCD of two numbers
```

```
    if (num1 < num2):
```

```
        return (find_gcd(num2, num1))
```

```
    elif (num1 % num2 == 0):
```

```
        return num2
```

```
    else:
```

```
        return ((find_gcd(num2, num1 % num2)))
```

```
def find_power(a, num1, num2):      # Find power of any number
```

```
    result = 1
```

```
    a = a % num2
```

```
    while(num1 > 0):
```

```
        if (num1 & 1):
```

```
            result = (result * a) % num2
```

```
            num1 = num1 // 2
```

```
            a = (a ** 2) % num2
```

```
    return result
```

```

def isprime(num):                                # Check whether the number is prime or not
    k = 10                                       # testing Variable
    if (num <= 1 or num == 4):
        return False
    if (num <= 3):
        return True
    while (k > 0):
        r = rn.randint(2, num-1)
        print(r)
        if (find_gcd(num, r) != 1):
            return False
        if (find_power(r, num - 1, num) != 1):
            return False
        k -= 1
    return True

if __name__ == "__main__":                      # Main function
    num = int(input("Enter the large number : "))
    if (isprime(num)):
        print(f"{num} is prime!")
    else:
        print(f"{num} is composite")

```

Output :-

```

12     elif (num1 % num2 == 0):
13         return num2
14     else:
15         return ((find_gcd(num2, num1 % num2)))
16
17 def find_power(a, num1, num2):
18     result = 1
19     a = a % num2
20     while(num1 > 0):
21         if (num1 & 1):
22             result = (result * a) % num2
23         num1 = num1 // 2
24         a = (a ** 2) % num2
25     return result
26
27 def isprime(num):
28     k = 10
29     if (num <= 1 or num == 4):
30         return False
31     if (num <= 3):
32         return True
33     while (k > 0):
34         r = rn.randint(2, num-1)
35         if (find_gcd(num, r) != 1):
36             return False
37         if (find_power(r, num - 1, num) != 1):
38             return False
39         k -= 1
40     return True
41
42 if __name__ == '__main__':

```

Variable explorer Help Files

Console 1/A

```

In [3]: runcell(0, 'D:/CLG2021/AA/LAB3/newpt.py')
Enter the large number :
521064401567922879406069432539095585333589848390805645835218385101837255573
5221
521064401567922879406069432539095585333589848390805645835218385101837255573
5221 is prime!
In [4]:

```

Python console History

```

18     result = 1
19     a = a % num2
20     while(num1 > 0):
21         if (num1 & 1):
22             result = (result * a) % num2
23         num1 = num1 // 2
24         a = (a ** 2) % num2
25     return result
26
27 def isprime(num):
28     k = 10
29     if (num <= 1 or num == 4):
30         return False
31     if (num <= 3):
32         return True
33     while (k > 0):
34         r = rn.randint(2, num-1)
35         if (find_gcd(num, r) != 1):
36             return False
37         if (find_power(r, num - 1, num) != 1):
38             return False
39         k -= 1
40     return True
41
42 if __name__ == '__main__':

```

Console 1/A

```

In [3]: runcell(0, 'D:/CLG2021/AA/LAB3/newpt.py')
Enter the large number :
521064401567922879406069432539095585333589848390805645835218385101837255573
5221
521064401567922879406069432539095585333589848390805645835218385101837255573
5221 is prime!
In [4]: runcell(0, 'D:/CLG2021/AA/LAB3/newpt.py')
Enter the large number : 123456789123456789123456789123456789123456789123456
123456789123456789123456789123456789123456789123456 is composite
In [5]:

```

Python console History

- About Algorithm :-

Time complexity of this solution is $O(k \log n)$. Note that power function takes $O(\log n)$ time.

Note that the above method may fail even if we increase number of iterations (higher k). There exist some composite numbers with the property that for every $a < n$, $\gcd(a, n) = 1$ and $a^{n-1} \equiv 1 \pmod{n}$. Such numbers are called Carmichael numbers. Fermat's primality test is often used if a rapid method is needed for filtering, for example in key generation phase of the RSA public key cryptographic algorithm.