Date:11.11.2022

**Final Year B. Tech., Sem VII 2022-23**

**Cryptography And Network Security Lab**

**Assignment submission**

**PRN : 2019BTECS00071**

**Name : Abhishek charpale**

**Batch: B3**

**Assignment: 9**

**Title of Assignment 9: Implementation of Prime Factorization of Coprime Number**

**Title:**

Implementation of Prime Factorization of Coprime Number

**Aim:**

To develop and implement Prime Factorization of Coprime Number

**Theory:**

* The RSA Factoring Challenge was a challenge put forward by [RSA Laboratories](https://en.wikipedia.org/wiki/RSA_Laboratories) on March 18, 1991[[1]](https://en.wikipedia.org/wiki/RSA_Factoring_Challenge#cite_note-1) to encourage research into [computational number theory](https://en.wikipedia.org/wiki/Computational_number_theory) and the practical difficulty of [factoring](https://en.wikipedia.org/wiki/Factorization) large [integers](https://en.wikipedia.org/wiki/Integer) and cracking [RSA](https://en.wikipedia.org/wiki/RSA_(algorithm)) keys used in [cryptography](https://en.wikipedia.org/wiki/Cryptography).
* They published a list of [semiprimes](https://en.wikipedia.org/wiki/Semiprime) (numbers with exactly two [prime factors](https://en.wikipedia.org/wiki/Prime_factor)) known as the [RSA numbers](https://en.wikipedia.org/wiki/RSA_numbers), with a cash prize for the successful factorization of some of them. The smallest of them, a 100-decimal digit number called [RSA-100](https://en.wikipedia.org/wiki/RSA-100) was factored by April 1, 1991. Many of the bigger numbers have still not been factored and are expected to remain unfactored for quite some time, however advances in [quantum computers](https://en.wikipedia.org/wiki/Quantum_computer) make this prediction uncertain due to [Shor's algorithm](https://en.wikipedia.org/wiki/Shor%27s_algorithm).
* The factoring challenge was intended to track the cutting edge in integer factorization. A primary application is for choosing the [key length](https://en.wikipedia.org/wiki/Key_length) of the [RSA](https://en.wikipedia.org/wiki/RSA_(algorithm)) [public-key encryption](https://en.wikipedia.org/wiki/Public-key_encryption) scheme. Progress in this challenge should give an insight into which [key sizes](https://en.wikipedia.org/wiki/Key_size) are still safe and for how long. As RSA Laboratories is a provider of RSA-based products, the challenge was used by them as an incentive for the academic community to attack the core of their solutions — in order to prove its strength.
* The RSA numbers were generated on a computer with no network connection of any kind. The computer's hard drive was subsequently destroyed so that no record would exist, anywhere, of the solution to the factoring challenge.

**Implementation of Prime Factorization of Coprime Number**

**Code:**

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

typedef vector<long long> vl;

string longDivision(string number, ll divisor)

{

// As result can be very large store it in string

string ans;

// Find prefix of number that is larger

// than divisor.

ll idx = 0;

ll temp = number[idx] - '0';

while (temp < divisor)

temp = temp \* 10 + (number[++idx] - '0');

// Repeatedly divide divisor with temp. After

// every division, update temp to include one

// more digit.

while (number.size() > idx) {

// Store result in answer i.e. temp / divisor

ans += (temp / divisor) + '0';

// Take next digit of number

temp = (temp % divisor) \* 10 + number[++idx] - '0';

}

// If divisor is greater than number

if (ans.length() == 0)

return "0";

// else return ans

return ans;

}

string multiply(string num1, string num2)

{

int len1 = num1.size();

int len2 = num2.size();

if (len1 == 0 || len2 == 0)

return "0";

// will keep the result number in vector

// in reverse order

vector<int> result(len1 + len2, 0);

// Below two indexes are used to find positions

// in result.

int i\_n1 = 0;

int i\_n2 = 0;

// Go from right to left in num1

for (int i = len1 - 1; i >= 0; i--)

{

int carry = 0;

int n1 = num1[i] - '0';

// To shift position to left after every

// multiplication of a digit in num2

i\_n2 = 0;

// Go from right to left in num2

for (int j = len2 - 1; j >= 0; j--)

{

// Take current digit of second number

int n2 = num2[j] - '0';

// Multiply with current digit of first number

// and add result to previously stored result

// at current position.

int sum = n1 \* n2 + result[i\_n1 + i\_n2] + carry;

// Carry for next iteration

carry = sum / 10;

// Store result

result[i\_n1 + i\_n2] = sum % 10;

i\_n2++;

}

// store carry in next cell

if (carry > 0)

result[i\_n1 + i\_n2] += carry;

// To shift position to left after every

// multiplication of a digit in num1.

i\_n1++;

}

// ignore '0's from the right

int i = result.size() - 1;

while (i >= 0 && result[i] == 0)

i--;

// If all were '0's - means either both or

// one of num1 or num2 were '0'

if (i == -1)

return "0";

// generate the result string

string s = "";

while (i >= 0)

s += to\_string(result[i--]);

return s;

}

ll isPrime(ll n)

{

// Corner case

if (n <= 1)

return 0;

// Check from 2 to square root of n

for (ll i = 2; i <= sqrt(n); i++)

if (n % i == 0)

return 0;

return 1;

}

int main()

{

ll t = 1;

//cin >> t;

string s;

cout<<"Enter the Number:\n";

cin >> s;

ll till = 100000;

for (ll i = 1; i < till; i++)

{

//cout << i << endl;

if (isPrime(i) == 0)

{

continue;

}

ll first = i;

string fs = to\_string(first);

string x = longDivision(s, i);

if (multiply(fs, x) != s)

continue;

cout << "\*\*" << endl;

cout << first << endl;

cout << x << endl;

cout << endl;

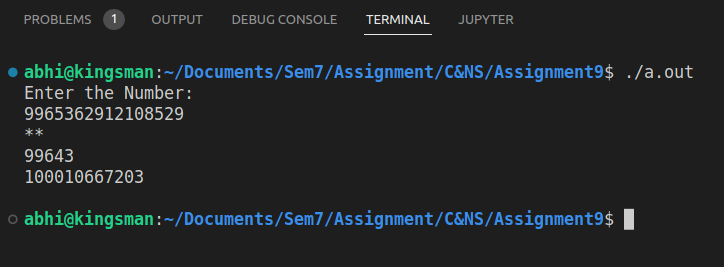
break;

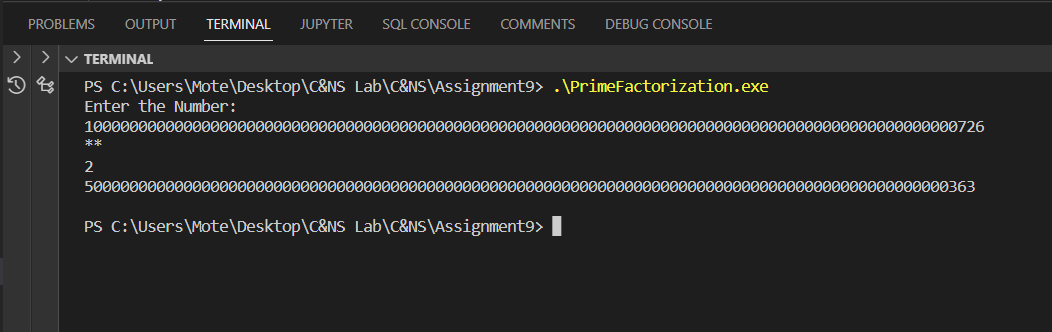
}

return 0;

}

**Output:**

****



**Conclusion:**

Performed the experiment successfully.