**Cryptography and Network Security Lab**

**Practical 7**

Q. Find the GCD of two given number using Euclidean Algo

**Theory:**

The Euclidean Algorithm for finding GCD(A,B) is as follows:

● If A = 0 then GCD(A,B)=B, since the GCD(0,B)=B, and we can

stop.

● If B = 0 then GCD(A,B)=A, since the GCD(A,0)=A, and we can

stop.

● Write A in quotient remainder form (A = B⋅Q + R)

● Find GCD(B,R) using the Euclidean Algorithm since GCD(A,B) =

GCD(B,R)

**Code:**

#include <bits/stdc++.h>

using namespace std;

int findGCD(int *num1*, int *num2*)

{

    if (*num2* == 0)

    return *num1*;

    cout<<*num1*/*num2*<<"\t"<<*num1*<<"\t"<<*num2*<<"\t"<<*num1*%*num2*<<endl;

    return findGCD(*num2*, *num1*%*num2*);

}

int main()

{

    int num1, num2;

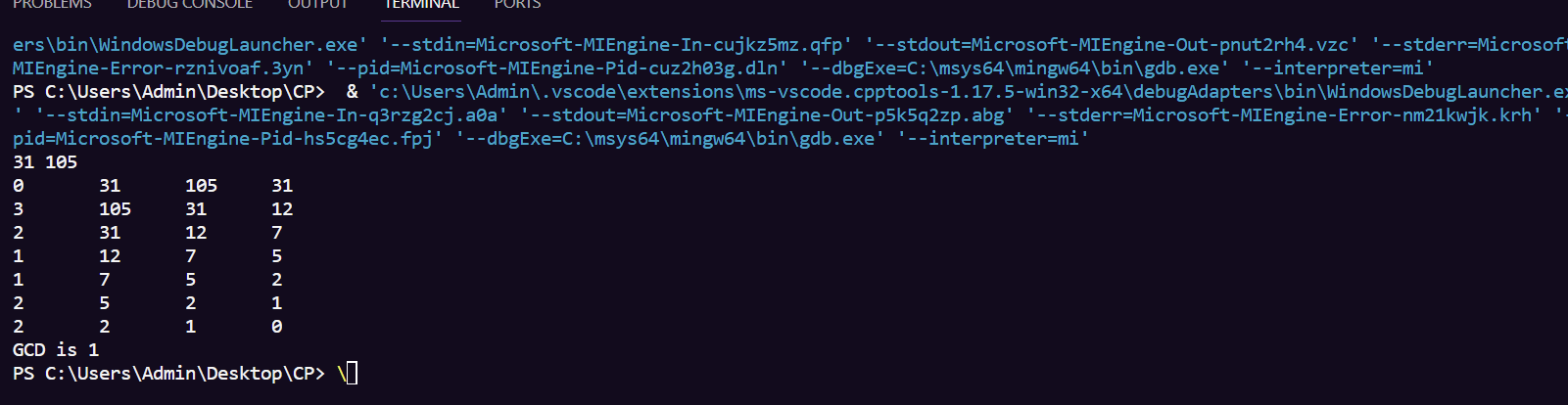
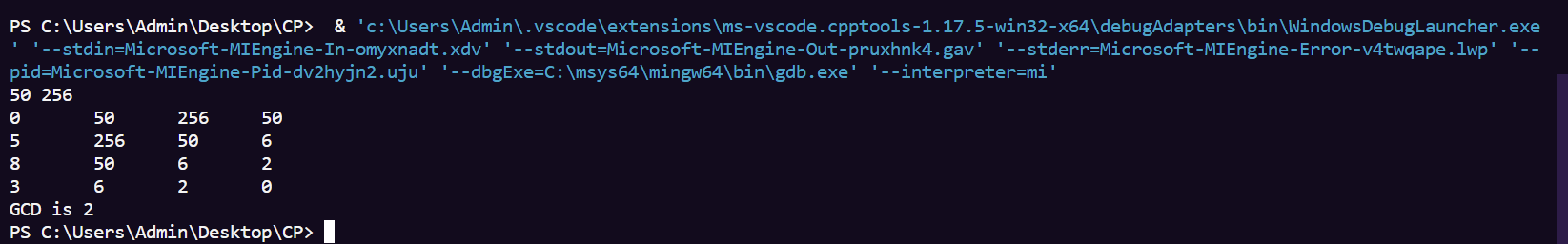
    cin >> num1 >> num2;

    int gcd = findGCD(num1, num2);

    cout << "GCD is " << gcd << endl;

    return 0;

}

**Results:**

Q. Prime Factorization of large numbers

**Theory:** We have to factorize a number such that its factors are prime and their product

equals a given number.

**Code:**

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

typedef vector<long long> vl;

#define pll pair<ll, ll>

#define vpll vector<pll>

#define vb vector<bool>

#define PB push\_back

#define MP make\_pair

#define ln "\n"

#define forn(*i*,*e*) for(ll i=0; i<e; i++

#define forsn(*i*,*s*,*e*) for(ll i=s; i<e; i++)

#define rforn(*i*,*e*) for(ll i=e; i>=0; i--)

#define rforsn(*i*,*s*,*e*) for(ll i=s; i>=e; i--)

#define vasort(*v*) sort(v.begin(), v.end())

#define vdsort(*v*) sort(v.begin(), v.end(),greater<ll>())

#define arrasort(*arr*,*n*) sort(arr,arr+n)

#define arrdsort(*arr*,*n*) sort(arr,arr+n,greater<ll>())

#define F first

#define S second

#define out1(*x1*) cout << x1 << ln

#define out2(*x1*,*x2*) cout << x1 << " " << x2 << ln

#define out3(*x1*,*x2*,*x3*) cout << x1 << " " << x2 << " " << x3 << ln

#define out4(*x1*,*x2*,*x3*,*x4*) cout << x1 << " " << x2 << " " << x3 << " " << x4 << ln

#define out5(*x1*,*x2*,*x3*,*x4*,*x5*) cout << x1 << " " << x2 << " " << x3 << " " << x4 << " " <<x5 << ln

#define out6(*x1*,*x2*,*x3*,*x4*,*x5*,*x6*) cout << x1 << " " << x2 << " " << x3 << " " << x4 << " "<< x5 << " " << x6 << ln

#define in1(*x1*) cin >> x1

#define in2(*x1*,*x2*) cin >> x1 >> x2

#define in3(*x1*,*x2*,*x3*) cin >> x1 >> x2 >> x3

#define in4(*x1*,*x2*,*x3*,*x4*) cin >> x1 >> x2 >> x3 >> x4

#define in5(*x1*,*x2*,*x3*,*x4*,*x5*) cin >> x1 >> x2 >> x3 >> x4 >> x5

#define in6(*x1*,*x2*,*x3*,*x4*,*x5*,*x6*) cin >> x1 >> x2 >> x3 >> x4 >> x5 >> x6

#define mz(*a*,*val*) memset(a,val,sizeof(a))

#define arrin(*a*,*n*) forn(i,n) cin >> a[i];

#define arrout(*a*,*n*) forn(i,n) {cout << a[i] << " ";} cout << ln;

#define fio ios\_base::sync\_with\_stdio(false);cin.tie(NULL);cout.tie(NULL)

#define mod 1000000007

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 += std::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;*

while (t--)

{

string s;

cin >> s;

ll till = 100000;

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

{

*//cout << i << endl;*

if (isPrime(i) == 0)

{

continue;

}

*//cout << i << endl;*

ll first = i;

string fs = to\_string(first);

string x = longDivision(s, i);

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

continue;

cout << first << endl;

cout << x << endl;

cout << endl;

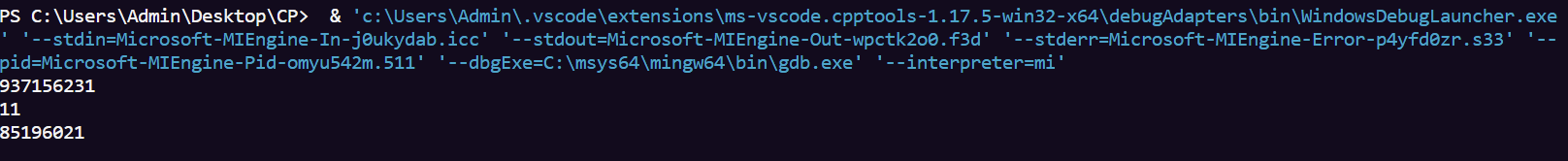
break;

}

}

return 0;

}

**Results:**