

#include<bits/stdc++.h>

using namespace std;

// Returns XOR of 'a' and 'b'

// (both of same length)

string xor1(string a, string b)

{

// Initialize result

string result = "";

int n = b.length();

// Traverse all bits, if bits are

// same, then XOR is 0, else 1

for(int i = 1; i < n; i++)

{

if (a[i] == b[i])

result += "0";

else

result += "1";

}

return result;

}

// Performs Modulo-2 division

string mod2div(string divident, string divisor)

{

// Number of bits to be XORed at a time.

int pick = divisor.length();

// Slicing the divident to appropriate

// length for particular step

string tmp = divident.substr(0, pick);

int n = divident.length();

while (pick < n)

{

if (tmp[0] == '1')

// Replace the divident by the result

// of XOR and pull 1 bit down

tmp = xor1(divisor, tmp) + divident[pick];

else

// If leftmost bit is '0'.

// If the leftmost bit of the dividend (or the

// part used in each step) is 0, the step cannot

// use the regular divisor; we need to use an

// all-0s divisor.

tmp = xor1(std::string(pick, '0'), tmp) +

divident[pick];

// Increment pick to move further

pick += 1;

}

// For the last n bits, we have to carry it out

// normally as increased value of pick will cause

// Index Out of Bounds.

if (tmp[0] == '1')

tmp = xor1(divisor, tmp);

else

tmp = xor1(std::string(pick, '0'), tmp);

return tmp;

}

// Function used at the sender side to encode

// data by appending remainder of modular division

// at the end of data.

void encodeData(string data, string key)

{

int l\_key = key.length();

// Appends n-1 zeroes at end of data

string appended\_data = (data +

std::string(

l\_key - 1, '0'));

string remainder = mod2div(appended\_data, key);

// Append remainder in the original data

string codeword = data + remainder;

cout << "Remainder : "<< remainder << "\n";

string b="000";

if(remainder == b)

cout<<"No Error Detected"<<endl;

else

cout<<"Error Detected"<<endl;

}

// Driver code

int main()

{

string data;

cout<<"Enter Data"<<endl;

cin>>data;

string key;

cout<<"Enter Polynomial"<<endl;

cin>>key;

encodeData(data, key);

return 0;

}