Isai Mercado Oliveros

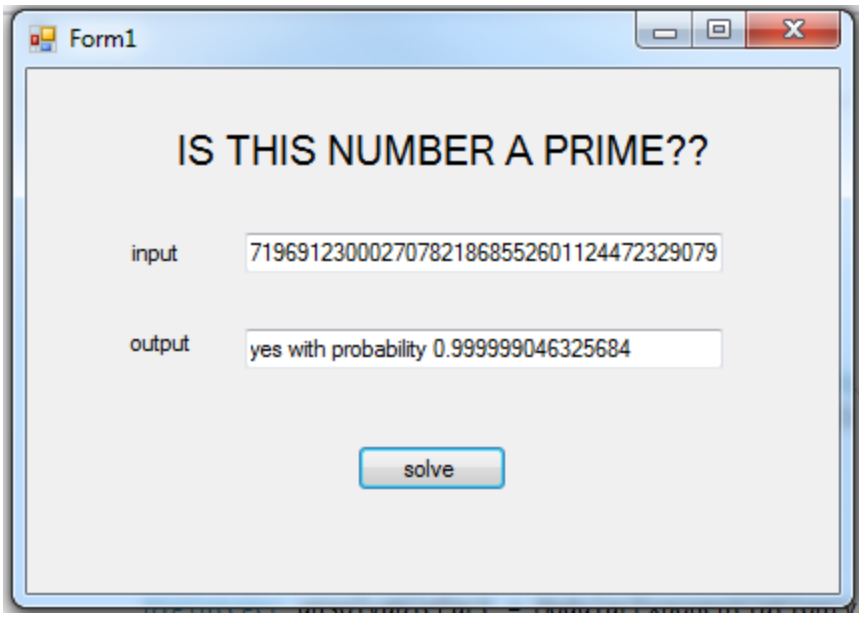
misaie

Section 1

Estimate of the amount of time you spent on this project at the top of the first

Spent 5 hours

Screenshot of application with a working example that is at least 50 decimal digits long.



Prime that is at least 50 decimal digits long by consulting the web. Be sure to give credit to your source.

229536868677196912300027078218685526011244723290 79

From

https://primes.utm.edu/lists/small/small.html

Equation used to compute the probability of *correctness* p that appears in the output.

probability = 1 - (1 / (Math.Pow(2, NUMBER_OF_TRIES)));

In other words 1 – 1 / 2 ^ 20

Implementation and application of modular exponentiation.

```
//     MODULAR EXPONENTIATION IMPLEMENTATION
private BigInteger ModularExponentiation(BigInteger value, BigInteger exponent, BigInteger modulo)
{

    if (exponent == 0)
    {
        return 1;
    }
    else if (mod(exponent, 2) == 0)
    {
        BigInteger result = ModularExponentiation(value, BigInteger.Divide(exponent, 2), modulo);
        BigInteger output = mod((BigInteger.Pow(result, 2)), modulo);
        return output;
    }
    else
    {
        BigInteger solvablePart = mod(value, modulo);
        BigInteger unsolvablePart = ModularExponentiation(value, BigInteger.Subtract(exponent, 1), modulo);
        BigInteger output = mod(BigInteger.Multiply(solvablePart, unsolvablePart), modulo);
        return output;
    }

}
```

Implementation and application of the Fermat primality tester.

```
//     FERMAT PRIMALITY TESTER
ModularExpResult = ModularExponentiation(randomNumber, possiblePrime - 1, possiblePrime);

if (ModularExpResult == 1)
{
    yesCounter++;
}
else
{
    noCounter++;
}
```

Remainder of your code.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Numerics;

namespace FermatLittleTheorem
{
    public partial class Form1 : Form
    {
        private const string YES = "yes with probability ";
        private const string NO = "no";
        private const int NUMBER_OF_TRIES = 20;

        private BigInteger possiblePrime;
        private BigInteger randomNumber;
        private double probability;
        private BigInteger ModularExpResult;

        private int yesCounter;
        private int noCounter;

        public Form1()
        {
            InitializeComponent();
        }

        private void solve_Click(object sender, EventArgs e)
        {
            //    INITIALIZING VARIABLES
            Random randomGenerator = new Random();

            possiblePrime = BigInteger.Parse(input.Text);

            randomNumber = 0;

            probability = 1 - (1 / (Math.Pow(2, NUMBER_OF_TRIES)));

            int logLength = Convert.ToInt32(BigInteger.Log(possiblePrime, 2) + 1);

            int length = logLength / 32;

```

```
49
50          //     SELECTING DIFFERENT RANDOM NUMBERS IF possiblePrime WAS LESS THAN 32 BITS OR NOT
51          if (length < 32)
52          {
53              while (randomNumber == 0)
54              {
55                  randomNumber = mod(randomGenerator.Next(), possiblePrime);
56              }
57          }
58          else
59          {
60              for (int i = 0; i < length; i++)
61              {
62                  randomNumber = (randomNumber << 32) + randomGenerator.Next();
63              }
64          }
65
66
67
68          //     DOING THE FERMAT TEST 20 TIMES TO IMPROVE PROBABILITY
69          for (int tries = 0; tries < NUMBER_OF_TRIES; tries++)
70          {
71              //     FERMAT PRIMALITY TESTER
72              ModularExpResult = ModularExponentiation(randomNumber, possiblePrime - 1, possiblePrime);
73
74              if (ModularExpResult == 1)
75              {
76                  yesCounter++;
77              }
78              else
79              {
80                  noCounter++;
81              }
82          }
83
84
85
86       // COMPARING COUNTER TO GIVE A SOLID ANSWER TO THE PROBLEM
87          if (yesCounter > noCounter)
88          {
89              output.Text = YES + probability.ToString();
90          }
91          else
92          {
93              output.Text = NO;
94          }
95
96      }
```

```
        //     MODULAR EXPONENTIATION IMPLEMENTATION
        private BigInteger ModularExponentiation(BigInteger value, BigInteger exponent, BigInteger modulo)
        {

            if (exponent == 0)
            {
                return 1;
            }
            else if (mod(exponent, 2) == 0)
            {
                BigInteger result = ModularExponentiation(value, BigInteger.Divide(exponent, 2), modulo);
                BigInteger output = mod((BigInteger.Pow(result, 2)), modulo);
                return output;
            }
            else
            {
                BigInteger solvablePart = mod(value, modulo);
                BigInteger unsolvablePart = ModularExponentiation(value, BigInteger.Subtract(exponent, 1), modulo);
                BigInteger output = mod(BigInteger.Multiply(solvablePart, unsolvablePart), modulo);
                return output;
            }

        }


        //     MOD FUNCTION
        private BigInteger mod(BigInteger number, BigInteger modulo)
        {
            BigInteger posResult = number % modulo;
            BigInteger negResult;

            if (posResult < 0)
            {
                negResult = BigInteger.Add(posResult, modulo);
                return negResult;
            }
            else
            {
                return posResult;
            }
        }
    }
}
```