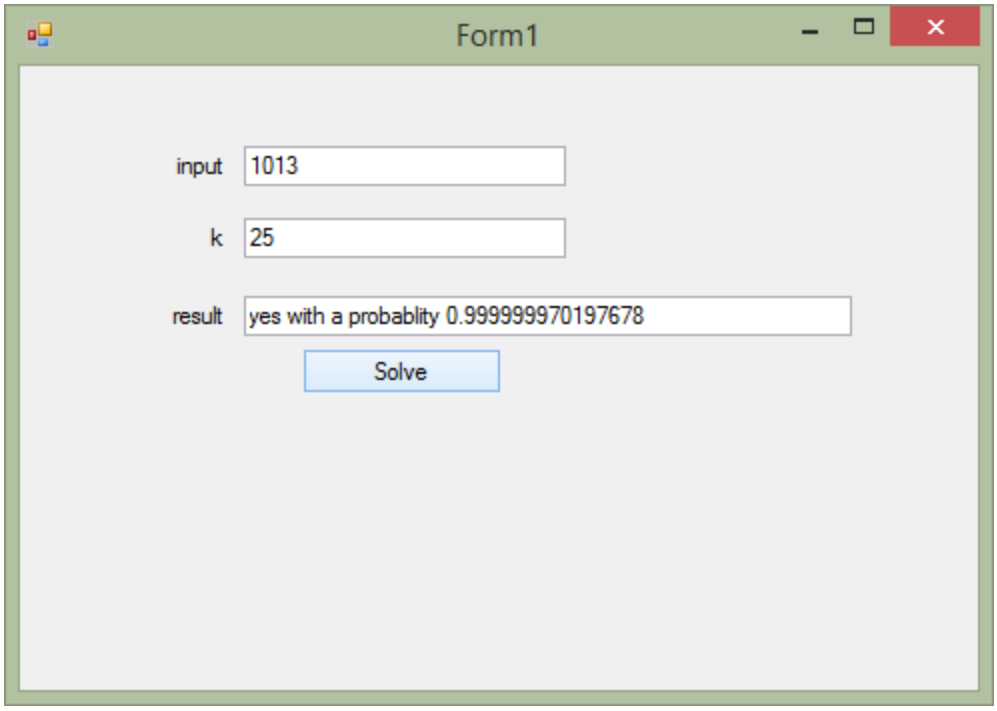
Project #1 Report
Kevin DeVocht
Section 002

Section 1: Screen Shot



Section 2: Code

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Numerics;
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;

namespace WindowsFormsApplication1
```

```
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

      private void solve_Click(object sender, EventArgs e)
      {
         //k is the number of times to run the primality tester to increase probabilty that a true prime has been
choosen.
         int K = Convert.ToInt32(k.Text);
         BigInteger N = BigInteger.Parse(input.Text);

                 for(int i = 0; i<K; i++)
         {
            //uniform generates a random number between 1 and N-1 for the primality tester to use
            BigInteger a = uniform(N-1);
            //use modulos exponentition to see if a and N are relativly prime
            if(modexp(a, N-1, N) == 1)
            {
               //Might be prime each time the loop reachs here it increases the probablity that
               // N is prime.
            }
            else
            {
               //If you ever reach here, you know that the number is not prime
               output.Text = "no";
               return;
            }
         }
         //If you have reached this point then you can be fairly certain that N
         // is prime or at the very least you can't prove that it is not prime
         // using the random numbers.
         decimal test = (decimal) (1 - (1 / (Math.Pow(2, K))));
         output.Text = "yes with a probablity " + test;
      }

      //Random Number Generator
      private BigInteger uniform (BigInteger N)
      {
         Random rand = new System.Random();
         int random_int = rand.Next();
         BigInteger random_number = 0;
         double length =  Math.Ceiling(BigInteger.Log(N, 2) + 1);
         for (int i = 0; i < length / 32; i++)
            random_number = (random_number << 32) + rand.Next();
```

```
        return (random_number % N);
    }

    //Modulos Exponentition
    private BigInteger modexp(BigInteger x, BigInteger y, BigInteger N)
    {
        if (y == 0)
        {
            return 1;
        }

        BigInteger r = 1;
        BigInteger z = x % N;
        z = modexp(x, (y/2), N);
        if(y.IsEven)
        {
            return (z*z) % N;
        }
        else
        {
            return x*(z*z) % N;
        }
    }

  }
}
```

Section 3: Equation

1-½^k