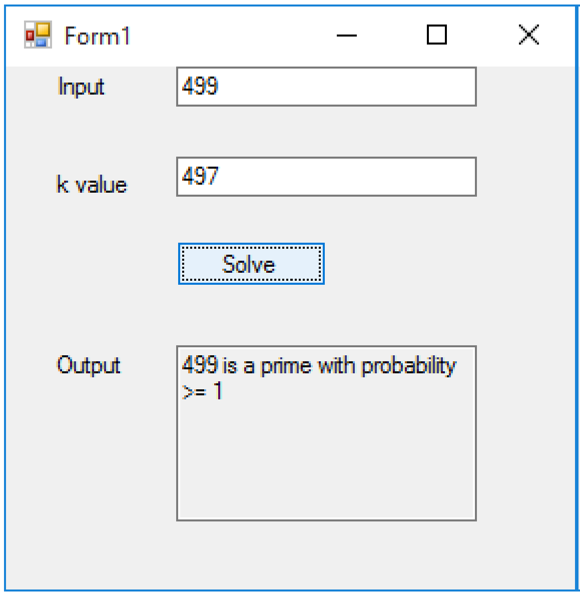
Anthony Constantino

15 September 2015

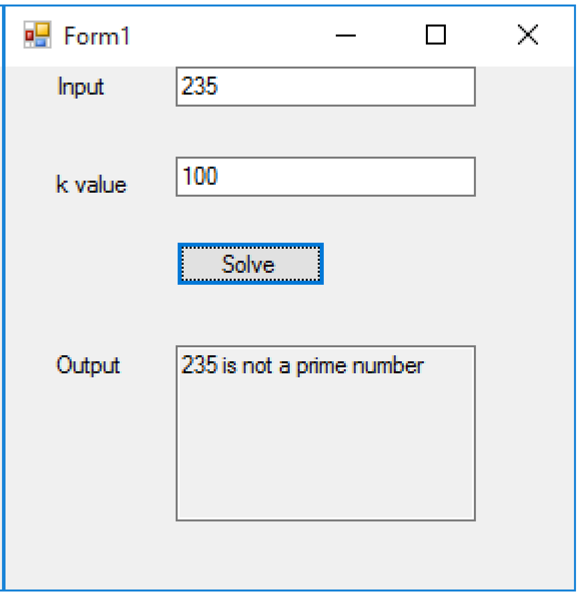
Project Submission



Input: 499

k value: 497

Output: 499 is a prime with probability >= 1



Input: 235

k value: 100

Output: 235 is not a prime number

The algorithm I have implemented to be the primality tester, begins by validating the input. Immediately afterwards, it beings a loop that it will go through k times (O(k)), based on the value that was passed in. In that loop it reuses the Random object to generate a new random number between 1 and (N – 1). Then beginning the modular exponentiation function (O(n3)). If the number is found to be prime it calculates the probability of the tests accuracy, if not it returns “not prime”.

```
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 Project1
{
    public partial class Form1 : Form
    {
        private Random rnd; //Private data member initialized once
        public Form1()
        {
            rnd = new Random();
            InitializeComponent();
        }

        //Calculates the greatest common divisor (Not used)
        public int gcd(int x, int y)
        {
            if (y == 0) return x;
            return gcd(y, x % y);
        }

       //Modular Exponentiation function O(n^3)
       //recursive portion will take n recursive calls
       //There is log y multiplications because the recursive call halves y each time
        public int modexp(int x, int y, int N)
        {
            if (y == 0) return 1;
            int z = modexp(x, y / 2, N);
            if (y % 2 == 0) return z * z % N;
            else return x * z * z % N;
        }

        //This function begins the primaility test
        private void SolveButton_Click(object sender, EventArgs e) //O(kn^3)
        {
            int input, k;
              //checks to make sure the inputs are numbers
            if (Int32.TryParse(this.InputBox.Text, out input)
               && Int32.TryParse(this.kValueBox.Text, out k))
            {
              //checks to make sure input and k are valid numbers to test primality
                if (input < 1 || k < 1 || k >= input)                 {
                    OutputBox.Text =
                            "Input must be positive integer and k must be 1 <= k < input";
                    return;
                }
                int probabilityK = k; //this is needed to calculate the probability
                for (; k > 0; k--) //We go through k number of times
                {
                    int a = rnd.Next(1, input - 1);
                    Console.WriteLine(a);
              //If at any point the result is not one we know it is not prime O(n^3)
              //If k != 0 we know we broke out of the for loop
                    if (modexp(a, input - 1, input) != 1) break;
                }

                if (k != 0) OutputBox.Text = input + " is not a prime number";
                else
                {
                     // Calculate the probability of the number being prime
                    double probability = (1 - 1 / Math.Pow(2, probabilityK));
OutputBox.Text = input + " is a prime with probability >= " + probability.ToString();
                }
            }
            Else
              //Error message if inputs are incorrect
                OutputBox.Text = "Either the input or k is not a number";

        }
    }
}
```