C# Quiz 23 — Tasks and Parallel Processing

C# Programming

This is a timed test. You have thirty minutes to complete the test. When you finish the test, upload your `Program.cs` to Canvas. Do not publish your answer to your `git` repository.

Consider the following console app. It computes all the prime numbers up to a user defined limit, times the computation, and prints the list of prime numbers and the amount of time it took to generate the list. If you run this with powers of 10 as imput (e.g., 10, 100, 1000, 10000, etc.) you will see that it rapidly reaches a point where it takes several minutes to produce output. The reason is in the method `foundP()`. This method finds prime numbers using brute force, that is, by checking every divisor of the number up to its square root. For input of 10000000 (ten million), it took about 17 seconds.

```
found primes in time 00:00:16.7341762
```

Parallelize this program to speed it up. On my machine for the same input of 10000000, it took about $2\frac{1}{2}$ seconds.

```
found primes in time 00:00:02.4195276
```

Here is the initial program. Do not attempt to optimize this by rewriting the method `foundP()`. Yes, there are more efficient ways of finding prime numbers. This quiz is designed to test your ability to write multiprocessing code.

```
using System;
using System.Collections.Generic;
using System.Diagnostics;

namespace csharp_Q23
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("C# quiz 23");
            Console.Write("Enter the highest number to find primes: ");
            string input = Console.ReadLine();
            long findTo = long.Parse(input);
            List<long> primes = new List<long>();

            Stopwatch sw = new Stopwatch();
            sw.Start();
            for (long i = 1; i < findTo; i++)
            {
                bool foundP = isPrime(i);
                //Console.WriteLine($"from foundP: i is {i}");
                if (foundP)
                {
                    primes.Add(i);
                }
            }
            sw.Stop();
            TimeSpan ts = sw.Elapsed;

            //output
            Console.WriteLine(”PRINTING␣PRIMES”);
            foreach (int element in primes)
                Console.Write($”␣{element}␣”);
            Console.WriteLine($”\n\nfound␣primes␣in␣time␣{ts}”);
        }

        private static bool isPrime(long n)
        {
            bool isP = true;
            for (int i = 2; i <= (int)Math.Sqrt(n); i++)
                if (n % i == 0)
                {
                    isP = false;
                    break;
                }
            return isP;
        }
    }
}
```