A screenshot of a computer screen

Description automatically generated

using System;

using System.Diagnostics;

using System.Collections;

using System.Collections.Generic;

namespace ECE2310\_HW06\_01a

{

class Program

{

static List<int> primes;

static void Main(string[] args)

{

// Initialize stopwatch for execution timing.

Stopwatch stopWatch = new Stopwatch();

stopWatch.Start();

Console.WriteLine("-----------------------------------------");

Console.WriteLine("Determining Perfect Numbers (using sieve)");

Console.WriteLine("-----------------------------------------");

// values for part a and part b

int aLimit = 2000;

int bStart = 5;

int bLimit = 5000;

// find max for only 1 list for both parts of program; more efficient

int limit = Math.Max(aLimit, bLimit);

/\*

Refer to wikipedia for "Sieve Of Eratosthenes"

Essentially creates a Look Up Table of primes up to limits

and stores them inside a List

\*/

BitArray bits = SieveOfEratosthenes(limit);

primes = new List<int>();

for (int i = 0; i < limit; i++)

{

if (bits[i])

{

primes.Add(i);

}

}

// Part B: Display perfect numbers and divisors from {1, 2000}

Console.WriteLine($"Perfect numbers between 1 and {aLimit}:");

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

{

if (isPerfect(i) == true)

{

Console.Write($"Perfect num: {i,-4} = ");

printDivisors(i);

Console.WriteLine();

}

}

// Part C: Display perfect numbers and divisors from {5, 5000}

Console.WriteLine("-----------------------------------------");

Console.WriteLine($"Perfect numbers between {bStart} and {bLimit}:");

int count = 0;

for (int i = bStart; i < 5000; i++)

{

if (isPerfect(i) == true)

{

count++;

Console.Write($"{i,-4}");

}

}

Console.WriteLine($"\nCount: {count}");

Console.WriteLine("-----------------------------------------");

stopWatch.Stop();

Console.WriteLine($"Execution Time: {stopWatch.ElapsedMilliseconds}ms");

}

private static bool isPerfect(int number)

{

for (int i = 0; i < primes.Count; i++)

{

/\*

Implementation of Euclid-Euler theorem

Theorem = 2^(p-1)\*[2^(p) - 1] is an even natural number when 2^(p-1) is prime

\*/

int x = (int)(Math.Pow(2, primes[i]) - 1);

long y = (long)(Math.Pow(2, primes[i] - 1) \* (Math.Pow(2, primes[i]) - 1));

if (y == number && primes.Contains(x))

{

return true;

}

}

return false;

}

private static void printDivisors(int number)

{

Console.Write("1");

for (int i = 2; i < number; i++)

{

if (number % i == 0)

{

Console.Write($" + {i}");

}

}

}

private static BitArray SieveOfEratosthenes(int limit)

{

/\*

Generates sieve using bitArry to limit range

Default value has all indexes of range set to true

THEN, it will go through and remove all multiples of numbers that are not prime.

Example: So it will go to 2, which is prime. THEN, it will set all

multiples of 2 (such as 4, 6, 8...) false aka not prime.

Then it will go to 3, which is prime. THEN, it will set all multiples of 3 (such as 15, 21...) false aka not prime. Note that when doing 3 it will not consider multiples of BOTH 2 and 3, such as 6, as 2 already set it false.

\*/

BitArray bits = new BitArray(limit + 1, true);

bits[0] = false;

bits[1] = false;

for (int i = 0; i \* i <= limit; i++)

{

if (bits[i])

{

for (int j = i \* i; j <= limit; j += i)

{

bits[j] = false;

}

}

}

return bits;

}

}

}