CSE1322L Assignment 7 - Spring 2024

Introduction:

For our last assignment this semester we are going to work with prime numbers. Prime numbers are interesting. A prime number is only divisible by itself and 1. The first prime number is 2, then 3, 5, 7, 11…

One method of finding prime numbers is called Sieve of Eratosthenes. For example, if we want to find all the primes below 20 we start off by writing all the numbers from 2 to 20.

2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Starting at 2, eliminate all it's multiples:

2	3	🟥 4	5	🟥 6	7	🟥 8	9	🟥 10	11	🟥 12	13	🟥 14	15	🟥 16	17	🟥 18	19	🟥 20

Now do the same for the next non-eliminated number (3):

2	3	🟥 4	5	🟥 6	7	🟥 8	🟥 9	🟥 10	11	🟥 12	13	🟥 14	🟥 15	🟥 16	17	🟥 18	19	🟥 20

Repeat up to the square root of 20 which is 4 in this case, so you are done. Thus the prime numbers below 20 are:
2, 3, 5, 7, 11, 13, 17, 19

For this assignment you'll first calculate all the primes lower than 10,000 and store them in a linked list.

Next you'll pick random numbers until you find a non-prime number. You'll know it's not prime because it will not be in the linked list of prime numbers.

Finally, you'll calculate which prime numbers can be multiplied together to make the random number. This is possible because all non-prime numbers are a product of prime numbers. For example if we were given 20 (non-prime), it's made up of 2x2x5 (all prime). If we were given 7710 (non-prime), it's made up of 257x5x3x2 (all prime).

Tasks:

- First we'll create a class called Prime.
  - If you are a java student import java.util.LinkedList, then create a linked list of Integers.
  - If you are a C# student, using System, System.Collections.Generic and Systems.Linq, then make a LinkedList of int.
  - Write a constructor which takes in an integer called max and calculates all primes up to max.
    - Start off with an array of booleans of the max size.
    - Initialize all cells to true
    - You are going to use this array of booleans to find the prime numbers. The index of the cell in the array represents the number you are deciding if it's prime or not. A true value in the cell indicates that index is prime, a false value indicates it's not. Follow the algorithm described above. You'll ignore cells 0 and 1 of the array, because those aren't prime numbers.
      - Using a loop, start at 2, and continue until you reach the square root of max
        - Change the value of each cell to false when you determine it's not a prime.
    - Once you've completed the algorithm all cells still true are prime.
    - Add all prime numbers to the linked list:
      - Java students, can use .add()
      - C# students can use .AddLast()
  - Write a method isPrime which takes in a number and returns true if it's found anywhere in the LinkedList.
    - Using a foreach loop, iterate over the LinkedList. If the number passed in is found in the list, return true, otherwise false.
  - Write a method getPrime which takes a position in the linked list and returns that value. For example, if passed 0 it should return 2 (the first prime), if passed 1 it would return 3 (the next prime), if passed 2 it would return 5 (the 3rd prime)...
    - Note:
      - Java students can use .get() on a linked list.
      - C# students will need .ElementAt(). ElementAt requires that you are using System.Linq

- Next in your Main class:
  - Create a constant and set it to 10000
  - Write a method findFactor which takes in a target (int) and a Prime object.
    - Use a loop starting at index 0 of the linked list and going up to the target.
      - Find the first prime number (using getPrime() in the Prime object) that evenly divides into the target. Return that number.
        - For example, if passed 10, it would return 2, since 2 is the first prime number that divides into 10 evenly.
        - If passed 9, it would return 3 since 2 doesn't divide evenly, but 3 does.

            - If passed 25, it'll return 5.
        - If you can't find a value, return -1 (this would only happen if the method were passed a number <2)
- In your main method:
    - Create an object of type Prime (the class you created above)
    - Create an object of type Random
    - Pick a random number lower than the constant.
        - If the number is <2 or is prime (you have a method for determining this) keep picking new numbers until you get a non-prime number>2.
    - Print out "What primes make up " followed by the number you picked.
    - Using a loop:
        - Call findFactor on the target number, which will return the lowest prime factor.
        - Print that factor (followed by the letter x, unless it's the last factor)
        - Divide the target by the returned factor
        - Repeat calling findFactor on the smaller values until you are left with a number less than 2.

Sample Output:

[NOTE: Each time it is run, you'll get a different random output]
[First Run]

```
What primes make up 7976?
997x2x2x2
```

[Second Run]

```
What primes make up 6638?
3319x2
```

[Third Run]

```
What primes make up 2860?
13x11x5x2x2
```

Submitting your answer:

Please follow the posted submission guidelines here:
https://ccse.kennesaw.edu/fye/submissionguidelines.php

Ensure you submit before the deadline listed on the lab schedule for CSE1322L here:
https://ccse.kennesaw.edu/fye/courseschedules.php

Optional task after you submit the working assignment:

You must use a linked list to store the prime values because we told you to (do not submit with anything else). However, in practice, this is actually a very inefficient way to do this algorithm. Retrieving a value from the middle of a linked list is O(n), whereas retrieving a value from the middle of an array is O(1).

If you'd like to see how much this can affect runtime, **after you submit your working assignment**, change the max value to 100,000,000, and attempt to run it as is. Note how long it takes to run. Now change the structure that stores the primes to an array and run it again, you'll see a dramatic improvement. Thus a linked list is actually not well suited for this task, an array is a better choice, especially since you know how many numbers you'll be holding. This is a lesson you should carry with you, don't randomly choose a data structure, think about which one is best suited to your task, and be open to trying alternative ones if you find your code is slow.

Again, do NOT submit your assignment with an array, you MUST use the linked list you were asked to, this is only for you to try after it's submitted successfully.