**Name: Session:**

**Programming I**

**Lab Exercise 11.12.2024**

**When you completed these programs, submit your source code.**

Today we will be implementing some algorithms developed by Greek mathematicians over 2000 years ago.

Eratosthenes of Cyrene was a Greek mathematician, geographer, poet, astronomer, and music theorist. He was the first person to use the word "geography" in Greek and he invented the discipline of geography as we understand it.

Euclid also known as Euclid of Alexandria, was a [Greek mathematician](http://en.wikipedia.org/wiki/Greek_mathematics), often referred to as the "Father of Geometry".

**Sieve of Eratosthenes**

In this activity we will use an old algorithm to find prime numbers as a more efficient alternative to check and guess.

1. Start IDLE and create a new window
2. Initialize three variables (numbers, primeList, and SIZE). numbers and primeList are lists. A list in Python is a structure that can hold a group of anything. In this case it will hold 1000 Boolean values.

numbers = [] #Create an empty list to hold Boolean values whose index is a prime candidate

primeList = [] #Create an empty list to hold prime numbers

SIZE = 1000

1. Initialize the list

for i in range(0, SIZE):

numbers.append(True)

1. Execute the sieve

for index in range(2, len(numbers)):

if numbers[index] == True:

#sets all elements of list after index to 0

#if it is a multiple of index

for i in range(index+1, len(numbers)):

if i % index == 0:

numbers[i] = 0;

1. Create a list of the prime numbers from 2 to SIZE (list elements that are still True)

for i in range(2, len(numbers)):

if numbers[i] == True:

primeList.append(i)

1. Now print out the list of prime numbers

print (primeList)

**Euclid’s Algorithm**

1. Write a function utilizing Euclid’s Algorithm to find the GCD of two user supplied integers. Here is a pseudocode implementation”

**Function** gcd(a, b)

**while** b ≠ 0

t := b

b := a **mod** b

a := t

**return** a

Once you have this working, we are going to design a Fraction class that uses the gcd function to reduce fractions.

1. Start by creating a new window in IDLE.
2. Add an \_\_init\_\_ method to the Fraction class

class Fraction:

def \_\_init\_\_(self, num, den):

self.num = num

if den != 0:

self.den = den

self.decimal = float(num)/den

self.reduce()

else:

print "Invalid fraction"

self.den = 1

1. Now add a reduce method to the Fraction class

def reduce(self):

div = self.gcd(self.num, self.den)

self.num = self.num / div

self.den = self.den / div

1. Now add the GCD function to the Fraction class

def gcd(self, a, b):

while b != 0:

t = b

b = a % b

a = t

return a

1. Finally add a \_\_str\_\_ method to the Fraction class

def \_\_str\_\_(self):

return str(self.num) + '/' + str(self.den)

1. You will now add the add, subtract, multiply, and divide methods to the Fraction class

def add(self, other):

n = self.num \* other.den + other.num \* self.den

d = self.den \* other.den

temp = Fraction(n, d)

return temp

def subtract(self, other):

TO DO: add code here

def multiply(self, other):

TO DO: add code here

def divide(self, other):

TO DO: add code here

1. Now test your code executing this block of code:

#Code to test the Fraction class

f1 = Fraction(1, 2)

f2 = Fraction(3, 4)

print ( f1)

print ( f2)

f3 = Fraction(6, 8)

print ( f3)

f4 = f1.multiply(f2)

f5 = f1.add(f2)

f6 = f1.subtract(f2)

f7 = f1.divide(f2)

print (f1, '\*', f2, '=', f4)

print( f1, '+', f2, '=', f5)

print (f1, '-', f2, '=', f6)

print (f1, '/', f2, '=', f7)

You should obtain the following output:

>>> ================================ RESTART ================================

>>>

1/2

3/4

3/4

1/2 \* 3/4 = 3/8

1/2 + 3/4 = 5/4

1/2 - 3/4 = -1/4

1/2 / 3/4 = 2/3

**Bacteria Growth**

The formula  can be used for estimating growth where:

y is the final amount

n is the initial amount

k is a constant

t is the time

For example, this formula could be used for estimating population growth in a region or for estimating cell growth in a lab experiment. Create a BacteriaGrowth application that calculates how many bacteria will be present based on this formula. The application should prompt the user for the initial bacteria, the constant k, and the time. Application output should look similar to:

Enter the initial bacteria amount: 5

Enter a constant value for k: 0.8

Enter growth time period in hours: 8

3009.2 bacteria will be present after 8.0 hours.