# Using task to launch independent processes

# The processes will do a countdown with a random delay

#

# Rodrigo Benavente

# 23/10/2021

defmodule Concur do

  @moduledoc """

  Functions to work with concurrent tasks

  """

  def prime(num) when num == 2 do #Number 2 is the first prime so we're adding a condition here

    boolL = true

  end

  def prime(num) do

    topLim = :math.sqrt(num) #Make the top limit the sqrt(num)

    topLim = round(topLim) #Round answer to create a list with bottom limit and top limit

    listS = Enum.to\_list(2..topLim) #Create a list with the limits per number

    boolL = for x <- listS do #Check if the num is prime or not and make a list

      if x < 2 do #If num is less than two return false

          false

        else if rem(num, x) == 0 do #Check if the number is divisible by "x" number inside the range

          false #if the num is divisible by"x" we add a false to the list

        else

          true #Else we add a true to the list

        end

      end

    end

    if Enum.member?(boolL, false) do #Check if the previous list indicates if "num" could be divided by a numer

      res = false #If it has a false then the number isn't prime

    else

      res = true #Else the number is prime

    end

  end

  def findPrime(number) do #Recieves a number and returns the sum of the prime numbers lower than it

    lista = Enum.to\_list(1..number-1)#Creates a lsit from 1 to "number - 1"

    numPrimos = for x <- lista do #Creates a list and makes iterates through the members of "lista"

      if prime(x) do #Calls the function prime for each member

        x

      else

        0

      end

    end

    sumaTot = Enum.sum(numPrimos)

  end

  def primeP(bottomL, topL) do #Recieves a bottom limit and a top limit

    lista = Enum.to\_list(bottomL..topL-1) #Creates a list from the bottom limit to the top limit

    numPrimos = for x <- lista do #Creates a list and makes iterates through the members of "lista"

      if prime(x) do #Calls the function prime for each member

        x #If it's a prime number we add it to the list

      else #We don't add anything to the list

        0

      end

    end

    sumaTot = Enum.sum(numPrimos) #Add all the prime numbers and return them

  end

  def parallelP(number) do # Recieves a number and returns the sum of the prime numbers lower than it

    cores = 8 #How many cores we're working with

    coresL = Enum.to\_list(1..cores) #Makes a list of [1,..,8]

    block = div(number, cores) #Creates a block

    bottomL = for y <- coresL do #The bottom limit of each block

      (y - 1) \* block

    end

    topL = for x <- coresL do #The top limit of each block

      if x == cores do

        number

      else

        (block \* x)

       end

    end

    result = 1..cores

      |> Enum.map(&Task.async(fn -> primeP(Enum.at(bottomL, (&1 - 1)), Enum.at(topL, (&1 - 1))) end)) #We call the parallel prime function

      |> Enum.map(&Task.await(&1, 50000)) #Wait for each tasks to end

      |> Enum.sum() #Sum all the results to get the total

  end

end