# Index

## A. Objetivos de la actividad :

i) Conocer los pros y contras de la computación paralela – multicore
ii) Tener un primer acercamiento a las posibles soluciones a estos problemas y los mecanismos que se utilizan para resolverlos en computación distribuida
iii) Aplicación práctica de la recursividad e ”iteratividad” en el contexto de la programación concurrente

## B. Enunciado de la Actividad Individual – Programación paralela :

1. Puntos previos que deberán seguirse :

i) Codificar en Python o Java (Recomendado usar Python ).
ii) Determinar la ganancia (tiempo) en segundos y/o milisegundos, del uso de la programación “tradicional” versus la programación paralela; similar al ejercicio realizado en la clase del 12/4/2020, adjunto link a la clase : https://eulti.bbcollab.com/recording/4bf10bea9ea04240a090bc0731c792f5 , también os podéis clonar el código en este link: https://github.com/sukuzhanay/programa-multicore-paralelo-recursivo.git

2. Para cada uno de los ejercicios siguientes (de la A -- C), se deberán aplicar las siguientes condiciones:

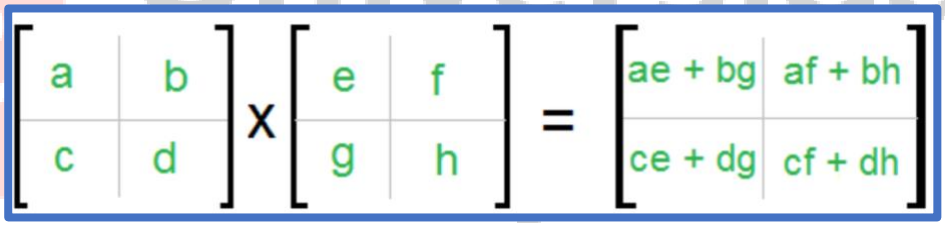
i) Para la codificación y correcta ejecución, él **n** ( número de datos ), será el número de expediente de cada alumno, ej. Christian Sucuzhanay: exp. 21535220 ( 21.535.220 millones).

ii) Deberá mostrarse una pantalla de Login, en la cual pida: email y contraseña; si es correcta ( si no; que imprima : No autorizado y salga ) que muestre el MENU principal, desde el que se deberá poder elegir el ejercicio a ejecutar.( Ver final del documento, Anexos)

iii) Todo el código deberá cumplir el siguiente formato: num_exp.py (21535220.py) y todos los ejercicios deberán estar bajo un mismo y único nombre de archivo (eje. : 21535220.py).

## Ejercicio A.

Codificar un programa que calcule el producto de dos matrices cuadradas de orden n^3 de números enteros, se deberá paralelizar el procesamiento en función del número de cores que tenga la máquina donde se ejecute el código.




## Ejercicio B.

Se pide codificar el algoritmo **mergeSort** para un array **n**, donde n es :

**n = vuestro numero de expediente.** ( En el ej. Usamos el exp. 21535220)

Se deberá generar el arrary n con 21.535.220 elementos de números enteros ( deberán ser aleatorios para poder ordenarlos ).

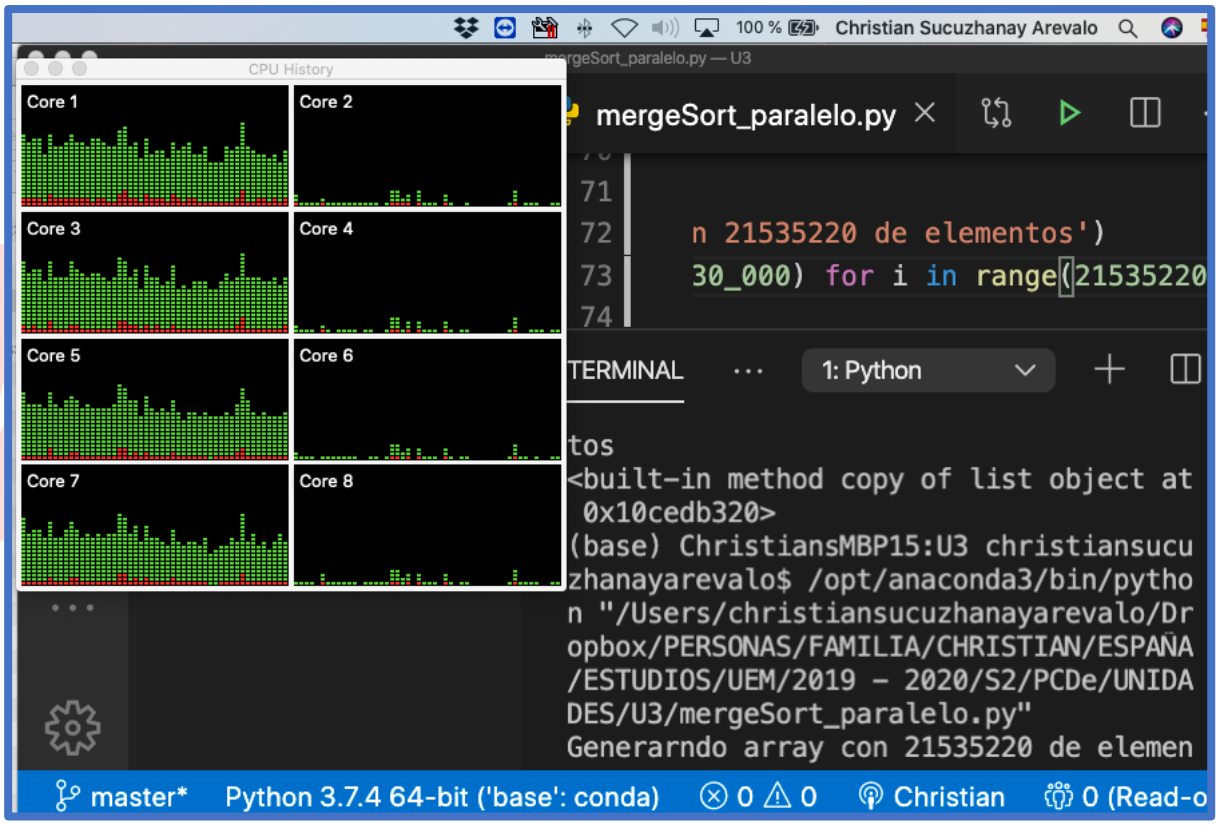
**Ej. mergeSort( 5 )**

RESULTADO = imprima el array ordenado [0,2,3,4,4]

### Ej. mergeSort( 21535220 )

RESULTADO = imprima el array ordenado [].

Es decir, se pide ordenar (21.535.220 millones de elementos) usando el algoritmo antes citado con la particularidad que se deberá **paralelizar** el procesamiento en función del número de cores que tenga la máquina donde se ejecute el código.



## Ejercicio C.

Se pide codificar un algoritmo que obtenga el numero **n** correspondiente a la secuencia de Fibonacci, donde n es : **n** = **vuestro numero de expediente.**

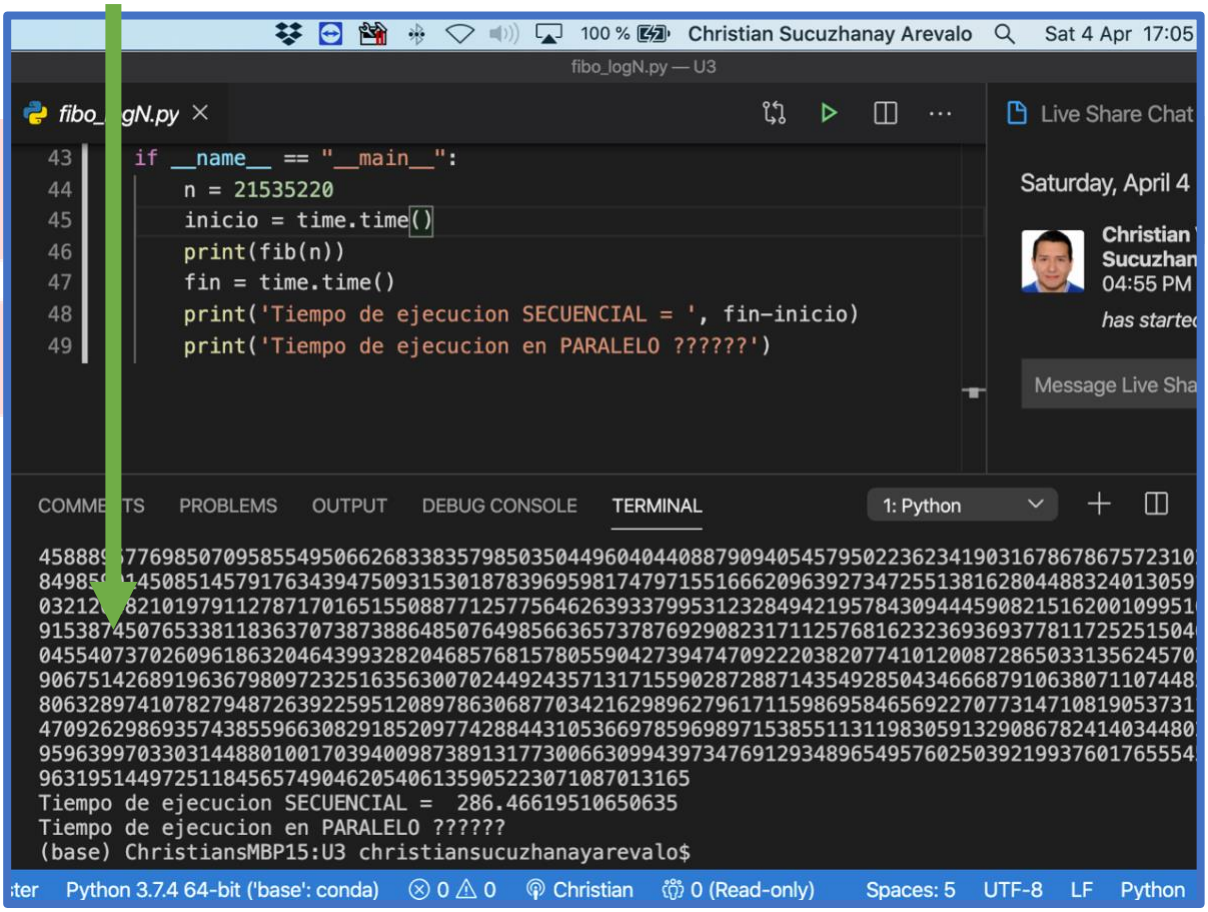
**Ej. fib( 215)**

RESULTADO:

382699285659014174244530850035136059033084785

**Ej. fib( 21535220 )**

RESULTADO




Con la particularidad que se deberá **paralelizar** el procesamiento en función del numero de cores que tenga la máquina donde se ejecute el código.

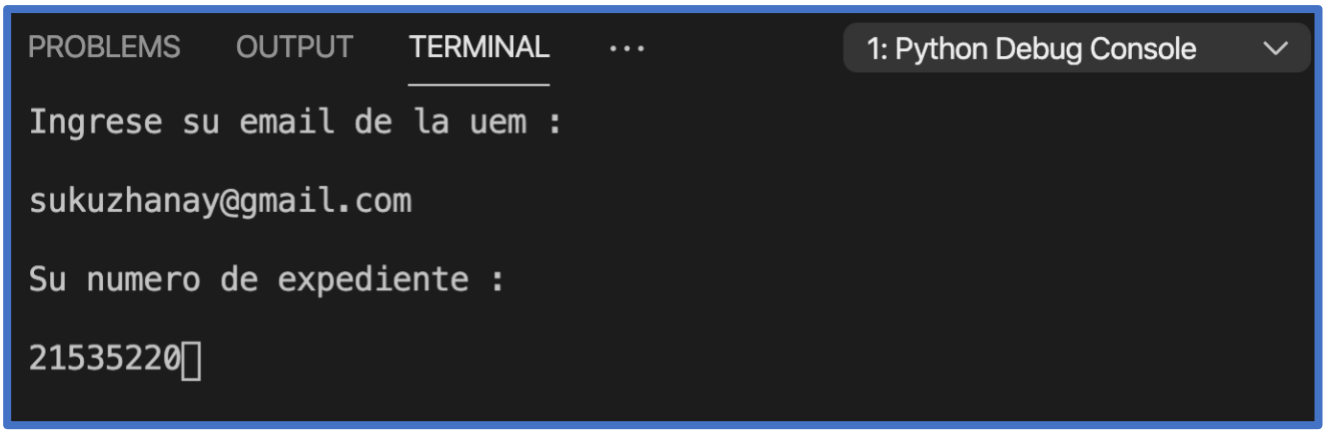
## C. Qué deberá entregar / subir al BlackBoard ? :

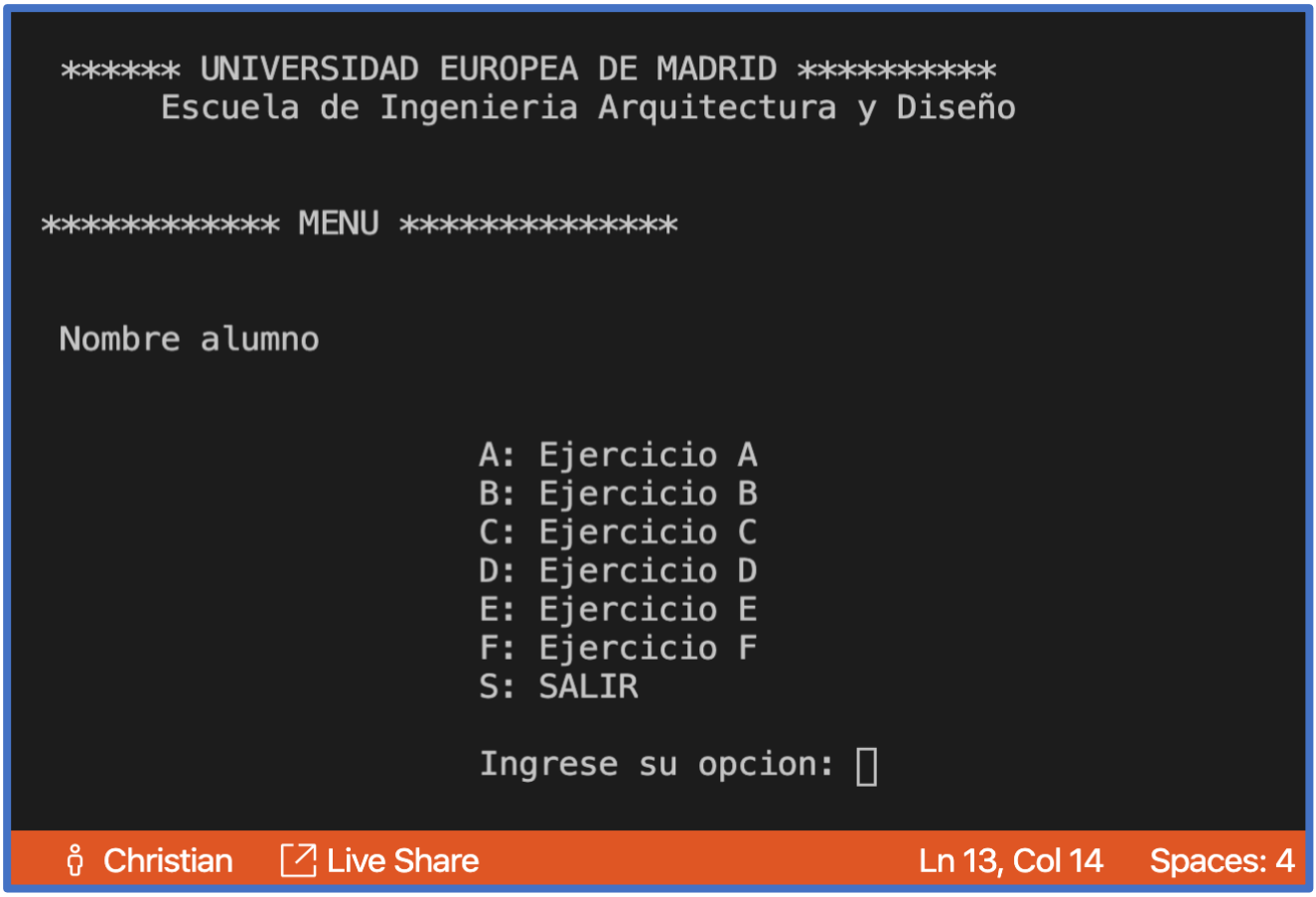
1. Memoria descriptiva de la investigación , presentación del problema, resolución, algoritmo ( solo en PDF ). **NO adjuntar el código en la memoria**, DEBE constar vuestro nombre de usuario y dirección de GITHUB (donde este el código.)
2. Se deberá explicar y comentar el código enviado.
3. Todo el código deberá subirse al GitHub de cada alumno y añadirme como colaborador ( para poder evaluar el código) al repositorio de GitHub; mi username es : sukuzhanay@gmail.com
4. La entrega es individual.
5. Fecha de entrega : 30 / 04 /2020 hasta las 23:59. (14 días)

## D. Denominación de archivo(s) e indicaciones de cómo subir y formato:

1. Todo(s) los archivos entregados deberán subirse de forma individual, NO comprimidos (zip, tar, etc)
2. Para nombrar cualquier archivo, deberá seguir la siguiente estructura:
   i. **Numexp.ext, ( por ej: 21535220.py)**

## E. Anexos




**Es esta última captura es SOLO referencial ( en la actividad se pide del A hasta el C)**