

EJERCICIOS PARA II EXAMEN PARCIAL.

1. Escriba una función llamada `partir(lista)` que reciba una lista y produce una lista de listas, donde cada sublista se forma con los números hasta encontrar la aparición de ceros, es decir, el cero va a ser el punto de corte para formar los números.

a. Utilizando recursividad de cola

b. Utilizando iteración

```
>>> partir([1,23,0,29,2,0,1,0,34])  
[[1, 23], [29, 2], [1], [34]]  
>>> partir([1,2,0,0,0,0,0])  
[[1, 2]]
```

2. Escriba una función **iterativa** `cambie_repetidos(num)` que reciba un número entero y sustituya todos los dígitos repetidos por 0 (excepto la primera ocurrencia del dígito repetido).

a. Utilizando recursividad de cola

b. Utilizando iteración

```
>>> cambie_repetidos(125315)  
125300  
>>> cambie_repetidos(1253467)  
1253467
```



3. Escriba una función **recursiva** `negativos(matriz)` que reciba una matriz de tamaño $m \times n$ y devuelva una tupla con las posiciones de la matriz (fila, columna) que contienen números negativos.

```
>>> negativos([-2, 1, 2, 4], [9, 8,-4,-3], [5,-8, 3, 2])  
[(0, 0), (1, 2), (1, 3), (2, 1)]
```

4. Escribir una función recursiva que recibe una matriz de dimensiones $n \times m$ (n = cantidad de filas, m = cantidad de columnas) de enteros y devuelve un vector con los máximos de cada una de las filas de la matriz.

```
>>> maximos([1, 2, 3], [8, 10, 1], [5, 6, 7])  
[3, 10, 7]
```

5. Escriba una función recursiva `producto_diagonal(matriz)` que reciba una matriz que debe ser $n \times n$ y que calcule el producto de la diagonal principal de una matriz cuadrada.

```
>>> producto_diagonal([[2, 3], [4, 5]])  
10
```

6. En una lista se reciben los salarios de n empleados de una empresa. Escriba una función `sueldos(lista)` que obtenga: (recursiva e iterativa)


- a) Cuántos empleados ganan más de 1 millón.
- b) Cuántos empleados ganan menos de 1 millón.
- c)Cuál es el salario más alto.
- d) El promedio de salarios.

```
>>> sueldos([482400, 1975600, 250000, 2675000])  
[2, 2, 2675000, 1345750]
```

7. Escriba una función recursiva `elimine_todos(ele, lista)` que reciba un elemento y una lista y elimine todas las apariciones del elemento en la lista, aún si el elemento se encuentra en una sublista (a cualquier nivel).

```
>>> elimine_todos(4, [4, 5, [[[3, 4], 5], [3]], 4, 8])  
[5, [[[3], 5], [3]], 8]
```

8. Escriba una función por medio de iteración `tenemas_par(num)` que recibe un número y retorna un valor boolean que indica si el número dado tiene más dígitos pares que impares. La función debe mostrar un comportamiento similar a los siguientes ejemplos:

<pre>>>> tienemas_par(237814934) False >>> tienemas_par(1357) False</pre>		<pre>>>> tienemas_par(222443) True >>> tienemas_par(12) False</pre>
---	---	---

