TEC | Tecnológico de Costa Rica

Instituto Tecnológico de Costa Rica **Área Académica de Ingeniería en Computadores** **Programa de Licenciatura en Ingeniería en Computadores** **Curso:** CE-1101 Introducción a la programación **Profesor:** Lic. Ing. Fabián Zamora Ramírez **Semestre:** I, 2018	**Práctica de temas:** • Programación iterativa con Python Para 3er examen.

INDICACIONES:

Para todos los siguientes ejercicios muestre su solución en lenguaje de programación Python. Toda función debe estar documentada con una descripción breve de lo que realiza y mostrar sus entradas (E), salidas (S) y restricciones (R). Si no se especifican restricciones, debe realizar las restricciones lógicas según el problema. Recuerde utilizar nombres significativos para sus funciones y variables. En caso de no cumplir con lo anterior se calificará que nota de cero.

1. Escriba una función **subvector(vec1, vec2)** que reciba como entrada 2 arreglos (con acceso a sus posiciones por medio de índices) y determine si el primer arreglo es un sub-arreglo del segundo. Ejemplos: si los vectores de entrada son [2, 3] y [1, 2, 3, 4, 5, 6, 7, 8, 9, 0] el resultado sería True. Si los vectores son: [1, 2, 3, 5] y [1, 2, 3, 4, 5, 6, 7, 8, 9, 0] el resultado sería False.
2. Escriba una función **unir(vector1, vector2)** que reciba como argumentos dos vectores del mismo tamaño y una ambos vectores. No se pueden utilizar los métodos append ni extend de listas.

   ```
   >>> unir(['a','b','c','d','e','f'], [1, 2, 3, 4, 5, 6])
   ['a','b','c','d','e','f', 1, 2, 3, 4, 5, 6]
   ```

3. Escriba una función **descomponer(vector)** que reciba un vector de tamaño 2 * n y obtenga dos nuevos vectores de tamaño n que contengan los elementos del vector que se envió como argumento, de forma que el primer elemento del vector va a ser el primero del primer vector resultante, el segundo elemento del vector va a ser el primero del segundo vector resultante y así sucesivamente.

   ```
   >>> descomponer([1, 3, 5, 7, 2, 4, 6 ])
   ([1, 5, 2, 6], [3, 7, 4])
   ```

4. Escribir una función **máximos(m)** que recibe una matriz de dimensiones nxm (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 iterativa **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]
   ```

6. Se define el doble factorial de un número n como:

$$n!! = \begin{cases} 1, & \text{si } n = 0 \text{ o } n = 1; \\ n \times (n-2)!! & \text{si } n \geq 2. \end{cases}$$

   Escriba una función iterativa llamada **doble_fact(n)** que calcule n!!. La función debe comportarse de la forma siguiente:

   ```
   >>> doble_fact(8)
   384
   ```

7. Escriba una función iterativa a llamar **iota(num)** que reciba un número natural y obtenga una lista en orden ascendente de números naturales menores al número dado.

   ```
   >>> iota(5)
   [0, 1, 2, 3, 4]
   >>> iota(1)
   [0]
   ```

8. Escriba una función iterativa **coincide(lista)** que recibe una lista de números enteros e indique si algún elemento de la lista coincide con la suma de todos los que le preceden.

   ```
   >>> coincide([2, 4, 3, 9, 14])  # 2 + 4 + 3 = 9
   True
   >>> coincide([5, 6, 14, 18, 20, 41])
   False
   ```

9. Escriba una función **mult_matrices(m1,m2)** que recibe y multiplique dos matrices.

10. Escriba una function **suma_matrices(m1,m2)** que recibe y sume dos matrices.

11. Escriba una function iterative **mult_escalar(m1,e1)** que recibe una matriz y la multiplica por un escalar.

12. Pueden encontrar más aqui: http://virtual.usalesiana.edu.bo/web/practica/archiv/p21.pdf

13. Pueden hacer todos los ejercicios anteriores pero utilizando iteración, en vez de recursividad.

OBJETOS.

14. Lista enlazada. Construya una lista enlazada con cuatro métodos:

a. **agregar(dato):** que agregar un elemento al final de la lista**.**
b. **insertar(dato):** que agrega un dato al inicio de la lista.
c. **removerInicio():** que remueve el primer elemento de la lista
d. **removerFinal():** que remueve el ultimo elemento de la lista

15. Diseñe e implemente un objeto tipo **Review**, que se va a utilizar para guardar reseñas sobre películas. Este objeto debe contener el nombre de la persona que hizo la reseña, el nombre de la película que se esta reseñando y la puntuación de 0 a 100.

a. Escriba una función iterativa **ordene(reviews)**, que reciba una lista con instancias de **Review** y las ordene utilizando el algoritmo burbuja de forma que la película con la mayor puntuación aparezca de primero y la película con la menor puntuación apareza de último.
b. Escriba una función iterativa **ordene_alreves(reviews)**, que reciba una lista con instancias de **Review** y las ordene utilizando el algoritmo burbuja de forma que la película con la mayor puntuación aparezca de último y la película con la menor puntuación apareza de primero.
c. Repita los ejercicios anteriores para los algoritmos **selección** e **inserción**.
d. Escriba una funcion iterativa **busque(reviews, nombre)**, que reciba una lista con instancias de Review y retorne True en caso de que se encuentre la pelicula con el nombre indicado en el segundo parametro. La busqueda se debe realizar de forma lineal.
e. Repita el ejercicio anterior utilizando busqueda binaria. Asuma que la lista viene desordenada.