ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

ALGORITMOS Y ESTRUCTURAS DE DATOS (TSDS)

ASIGNATURA: ALGORITMOS Y ESTRUCTURAS DE DATOS

PROFESOR: Ing. Lorena Chulde

PERÍODO ACADÉMICO: 2024-B

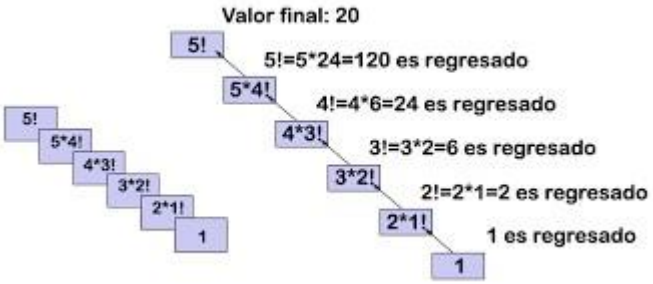
TAREA Grupal

TÍTULO:

FUNCIONES RECURSIVAS

Nombres de los estudiantes:

Tapia Alan y Zapata Felipe



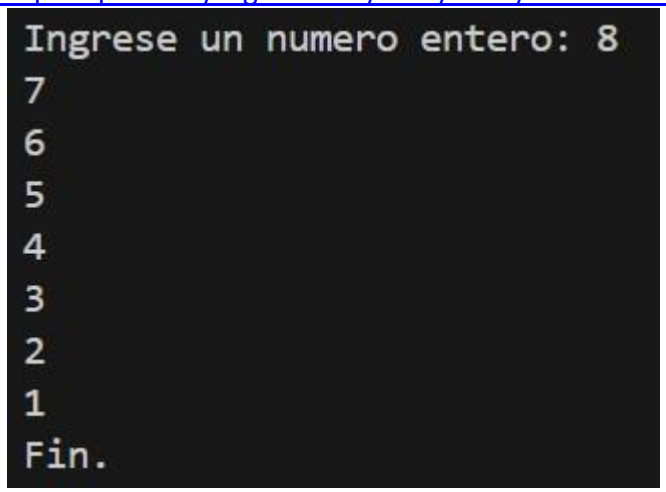
2024-B

PROPÓSITO DE LA TAREA

Reutilizar el código mediante funciones recursivas para una programación óptima.

Parte I: Funciones recursivas

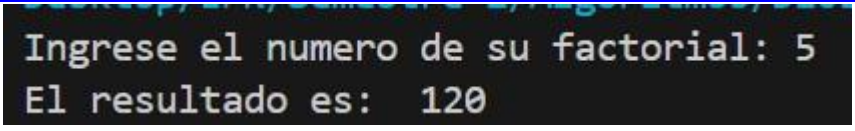
1. Implementar una función recursiva que reciba un parámetro de tipo entero y luego llame en forma recursiva con el valor del parámetro menos 1.
   https://github.com/FelipeZapata137/Algoritmos-/blob/main/Parcial%203/D10E01.py



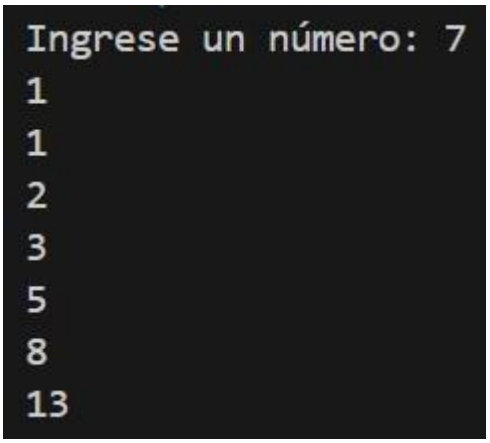
2. Implementar una función recursiva que imprima en forma descendente de 5 a 1 de uno en uno.
   https://github.com/FelipeZapata137/Algoritmos-/blob/main/Parcial%203/D10E02.py

3. Usando funciones recursivas, calcular el factorial de un número n!.

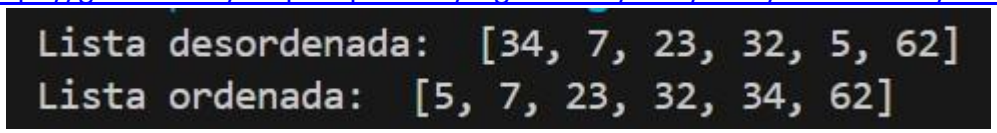
   El factorial de un número n se define como la multiplicación de todos sus números predecesores hasta llegar a uno. Por lo tanto 5!, leído como cinco factorial, sería 5*4*3*2*1.
   https://github.com/FelipeZapata137/Algoritmos-/blob/main/Parcial%203/D10E03.py



4. Usando funciones recursivas, calcular la serie de fibonacci. Dicha serie calcula el elemento n sumando los dos anteriores n-1 + n-2. Se asume que los dos primeros elementos son 0 y 1.
https://github.com/FelipeZapata137/Algoritmos-/blob/main/Parcial%203/D10E04.py



5. Implementar un método recursivo para ordenar los elementos de una lista.
https://github.com/FelipeZapata137/Algoritmos-/blob/main/Parcial%203/D10E05.py



Parte II: Consulta de Algoritmos

Consultar y realizar una presentación sobre los siguientes algoritmos:

- Algoritmos de ordenamiento (inserción sort)

```
def insertion_sort(arr):
    # Recorremos desde el segundo elemento hasta el final
    for i in range(1, len(arr)):
        # Guardamos el valor actual y el índice anterior
        current_value = arr[i]
        j = i - 1

        # Desplazamos los elementos que sean mayores al valor actual
        while j >= 0 and arr[j] > current_value:
            arr[j + 1] = arr[j]
            j -= 1

        # Insertamos el valor actual en su posición correcta
        arr[j + 1] = current_value

    return arr

# Ejemplo de uso
numeros = [5, 3, 8, 6, 2]
print("Arreglo original:", numeros)
print("Arreglo ordenado:", insertion_sort(numeros))
```

- Algoritmos de ordenamiento (quick sort)

```
def quick_sort(array):
    # Caso base: Si el array tiene 0 o 1 elemento,
ya está ordenado
    if len(array) <= 1:
        return array

    # Elegir un pivote (normalmente el último
elemento)
    pivot = array[-1]

    # Dividir en listas menores y mayores al pivote
    less_than_pivot = [x for x in array[:-1] if x <=
pivot]
    greater_than_pivot = [x for x in array[:-1] if x
> pivot]

    # Ordenar recursivamente y combinar
    return quick_sort(less_than_pivot) + [pivot] +
quick_sort(greater_than_pivot)


# Ejemplo de uso
if __name__ == "__main__":
    unsorted_list = [10, 7, 8, 1, 6, 3, 5]
    sorted_list = quick_sort(unsorted_list)
    print("Lista ordenada:", sorted_list)
```