	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: Adrian Ulises Mercado Martinez

Asignatura: Estructura de Datos y Algoritmos I

Grupo: 13

No de Práctica(s): 12

Integrante(s): Méndez Bernal Luis Alberto

No. de Equipo de cómputo empleado:

No. de Lista o Brigada: 13

Semestre: 2020-2

Fecha de entrega: 7 de Junio del 2020

Observaciones:

CALIFICACIÓN: _____

INTRODUCCION

La práctica tiene el nombre de “Recursividad” esto se refiere al hecho de que un problema se ira repitiendo hasta hacerse muy muy simple. Podemos decir que es llamar a la misma función varias veces para que esta siga trabajando hasta que lleguemos a un valor que nos interese.

DESARROLLO

FACTORIAL

Para el desarrollo empezaremos haciendo un programa que nos calcule la factorial de un numero dado. La formula de las factoriales es

$$n! = \prod_{i=1}^{n} p_i = 1 \times 2 \times 3 \ldots \times (n-1) \times n$$

Sabiendo eso podemos crear el programa

```
1
2   import math
3   def factorial(n):
4       if n<2:
5           return 1
6
7       return n * factorial(n-1)
8
9   num = int(input("Ingrese un número\t"))
10  print("El Factorial de", num, "es", factorial(num))
```

Y su ejemplo es:

```
factorial(5)
120
```

1*2=2 2*3=6 6*4=24 24*5=120

HUELLAS DE TORTUGA

Para este programa usaremos la biblioteca Turtle, este consiste en marcar el recorrido de una tortuga en espiral, haciendo marcas más separadas conforme pase el “tiempo”

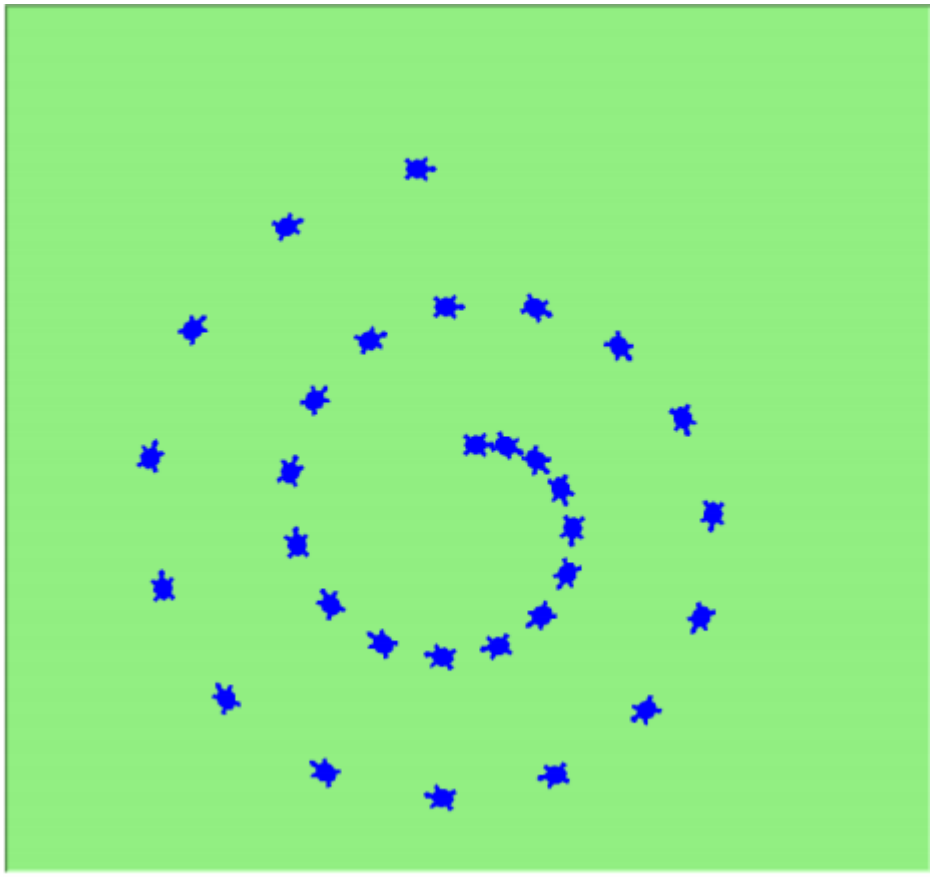
```
import turtle

def recorrido_recursivo(tortuga, espacio, huella):
    if huella > 0:
        tortuga.stamp()
        espacio = espacio + 3
        tortuga.forward(espacio)
        tortuga.right(24)
        recorrido_recursivo(tortuga, espacio, huella - 1)

wn = turtle.screen()
wn.bgcolor("lightgreen")
wn.title("Tortuga")
tess = turtle.Turtle()
tess.shape("turtle")
tess.color=("blue")
tess.penup()
recorrido_recursivo(tess, 20, 30)

wn.mainloop()
```

Y lo que hace es esto:



FIBONACCI

Usando el nuevo método de recursividad creamos un nuevo código para la sucesión de Fibonacci

```python
import math
def fibonacci(n):
    Memoria = [0, 1]

    while len(Memoria) < n + 1:
        Memoria.append(0)

    if n <= 1:
        return n
    else:
        if Memoria[n - 1] == 0:
            Memoria[n - 1] = fibonacci(n - 1)

        if Memoria[n - 2] == 0:
            Memoria[n - 2] = fibonacci(n - 2)

    Memoria[n] = Memoria[n - 2] + Memoria[n - 1]
    return Memoria[n]

num = int(input("Ingrese un número\t"))
print("El Numero de Fibonacci de", num, "es", fibonacci(num))
```

CONCLUSION

La recursividad nos puede ayudar a facilitar tareas que tengan que ver con repeticiones, ya que, en vez de escribirlo todas esas veces, el programa retoma la función y la vuelve a activar hasta que le digamos. Uno de los problemas que capte es que es medio difícil deducir la función base de algunos programas, y esto puede hacer que nos retrasemos bastantito. Otra cosa que note es que tarda un poco en ejecutarse el programa ya que tiene que repetirse muchas veces y esto gasta tiempo y memoria