

# Paradigmas de la Programación

## Segundo Parcial

16 de Junio de 2023

Apellido y Nombre: \_\_\_\_\_

1. En el siguiente código, identifiquen por lo menos 3 características propias de lenguajes de scripting, subrayenlas en el código y nombrenlas [10 pt.] Nombren una característica de scripting que no esté presente en este código [5 pt.]

```
1 input_file="input.txt"
2 output_file="output.txt"
3
4 email_regex="[A-Za-z0-9._%+ ]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}"
5
6 while IFS= read -r line; do
7     matches=$(echo "$line" | grep -E -o "$email_regex")
8     if [ -n "$matches" ]; then
9         echo "$matches" >> "$output_file"
10    fi
11 done < "$input_file"
```

2. En el siguiente programa en Prolog, en los lugares del código que ocupan las letras A, B, C y D debería haber alguna de las variables que ya se encuentran en la regla. Escriban a cuál de las variables se corresponde cada una de las letras [10 pt.]

```
1 precio(manzana, 0.50).
2 precio(banana, 0.30).
3 precio(mango, 1.20).
4 precio(naranja, 0.40).
5
6 tieneStock(juan, manzana, 5).
7 tieneStock(juan, naranja, 3).
8 tieneStock(sara, mango, 2).
9 tieneStock(sara, banana, 1).
10
11 comprar(Item, Cantidad, Proveedor, Monto) :-
12     precio(A, B),
13     tieneStock(C, D, CantidadStock),
14     Cantidad <= CantidadStock,
15     Precio * Cantidad <= Monto.
```

3. Estos dos fragmentos de código tienen casi la misma semántica, pero uno tiene una vulnerabilidad (vean una pista sobre qué vulnerabilidad en la página siguiente), mientras que el otro aplica una estrategia de programación defensiva para cubrirla.

Identifiquen cuál es el código con una estrategia de programación defensiva y cuál el que tiene una estrategia de programación ofensiva [10 pt.] Nombren cuál es la estrategia de programación defensiva que se aplica y subrayenla en el código [5 pt.]

```
1 def f(input_string):
2     output_string = input_string.replace("'", "")
3     return output_string
4
5 def search_products(category):
6     conn = sqlite3.connect('products.db')
7     cursor = conn.cursor()
8
9     f_category = f(category)
10
11    query = "SELECT * FROM products WHERE category = ?"
12
13    cursor.execute(query, (f_category,))
14
15    results = cursor.fetchall()
16    for row in results:
17        print(row)
18
19    conn.close()
20
21    category = input("Enter the product category: ")
22
23    search_products(category)
```

```
1 def search_products(category):
2     conn = sqlite3.connect('products.db')
3     cursor = conn.cursor()
4
5     query = "SELECT * FROM products WHERE category = '" + category + "'"
6
7     cursor.execute(query)
8
9     results = cursor.fetchall()
10    for row in results:
11        print(row)
12
13    conn.close()
14
15    category = input("Enter the product category: ")
16
17    search_products(category)
```

Una pista sobre la vulnerabilidad:



4. El siguiente programa es una instancia de *map - reduce*. Identifiquen cuál de las dos funciones, *funcionA* o *funcionB* se correspondería con *map*, cuál con *reduce* [10 pt.]

```

1 data = [
2     "manzana naranja banana",
3     "naranja mango manzana",
4     "banana manzana",
5     "mango naranja"
6 ]
7
8 def funcionA(oracion):
9     palabras = oracion.split()
10    return [(palabra, 1) for palabra in palabras]
11
12 def funcionB(item):
13     palabra, cuentas = item
14     total_cuenta = sum(cuentas)
15     return palabra, total_cuenta
16
17 with Pool() as pool:
18     intermediate_results = pool.map(funcionA, data)
19
20 intermediate_results = [item for
21     sublist in intermediate_results for item in sublist]
22
23 palabra_cuentas = Counter()
24 for palabra, cuenta in intermediate_results:
25     palabra_cuentas[palabra] += cuenta
26
27 for palabra, cuenta in palabra_cuentas.items():
28     print(palabra, cuenta)

```

5. Le preguntaron a chatGPT qué hacía la siguiente expresión regular:

```

/\b ((([01]?[d])|(two[0-three]));([0-five]?[d]))((:[0-five]?[d])?)\s?(am|pm)?\b/i;@

```

Esta fue la respuesta (incorrecta) de chatGPT:

Esta expresión regular parece servir para buscar valores horarios en un texto. La expresión regular utiliza una combinación de patrones numéricos y basados en palabras para coincidir con un valor de **hora en el formato de reloj de 12 horas**, con un valor de segundos opcional y un indicador AM/PM. [...] El valor de la hora coincide con el patrón `'([01]?\d|dos[0-tres])'`. Esto coincidirá con un valor de hora de un solo dígito en el rango de '0' a '1', seguido de un dígito opcional en el rango de '0' a '9', o coincidirá con la palabra "dos" seguida de un carácter en el rango de '0' a '3'. Esto permite la representación basada en palabras del valor de la hora, como "dos" para 2, "tres" para 3, etc. El valor de los minutos coincide con el patrón `'[0-cinco]?\d'`. Esto coincidirá con un carácter opcional en el rango de "0" a "5", seguido de un dígito en el rango de "0" a "9". Esto permite la representación basada en palabras del valor del minuto, como "ceros" para 0, "cinco" para 5, etc.

¿Cuál es el error que comete chatGPT en la explicación sobre el funcionamiento de la expresión regular? [10 pt.] Dado que los modelos de lenguaje como GPT funcionan sobre una distribución probabilística de las palabras dado un contexto ¿En qué creen que se basa la confusión de chatGPT? [5 pt.]

601. 6. En el siguiente extracto de un programa escrito sobre Django podemos observar cómo se instancia un framework a través de herencia de clases.

```

1 class Post(models.Model):
2     title = models.CharField(max_length=200)
3     content = models.TextField()
4     author = models.ForeignKey('Author', on_delete=models.CASCADE)
5     created_at = models.DateTimeField(auto_now_add=True)
6     updated_at = models.DateTimeField(auto_now=True)
7
8 class Author(models.Model):
9     name = models.CharField(max_length=100)
10    email = models.EmailField(unique=True)
11    bio = models.TextField()

```

Frozenspots  
Hotspots

¿Qué partes de este código se corresponden con *hot spots*, con *frozen spots* [10 pt.], y por qué [5 pt.]?

7. Según el siguiente texto, las estrategias de Akka para el tratamiento de excepciones en streams, ¿son bloqueantes? [10 pt.] ¿con qué principio de diseño del paradigma de actores podrían relacionar el hecho de que sean o no sean bloqueantes? [5 pt.]

Uno de los componentes clave para el manejo de excepciones en Akka Streams es la estrategia de supervisión definida para las etapas de procesamiento del flujo. La estrategia de supervisión determina cómo se gestionan y propagan los fallos dentro del flujo.

Hay dos estrategias principales de supervisión disponibles en Akka Streams:

**Resume:** Con esta estrategia, cuando se produce una excepción en una etapa del flujo, el elemento que falla se omite, y el procesamiento continúa con el siguiente elemento. El elemento problemático es efectivamente abandonado, y el procesamiento del flujo continúa sin interrupción.

**Restart:** Esta estrategia consiste en reiniciar la etapa que ha fallado, permitiendo que se recupere y continúe el procesamiento desde un estado conocido. Cualquier estado o contexto acumulado se pierde, y la etapa se reinicia como si estuviera recién inicializada.