Paradigmas de la Programación – Tercer Parcial

10 de Junio de 2025

Apellido y Nombre: _____

Ej. 1 Ej. 2 Ej. 3 Ej. 4 Ej. 5 Ej. 6 Ej. 7

1. Dada la siguiente base de conocimiento:

```
1  requisito(familiar, infantil).
2  requisito(familiar, grupal).
3
4  requisito(de_viaje, no_fragil).
5  requisito(de_viaje, robusto).
6  requisito(de_viaje, compacto).
7
8  aprueba(Juego, infantil) :-
9      not(violento(Juego)),
10     simple(Juego).
11 aprueba(Juego, grupal) :-
12     jugadores(Juego,N),
13     N>2.
14 aprueba(Juego, robusto) :-
15     magnetico(Juego).
16 aprueba(Juego, robusto) :-
17     encastrable(Juego).
18 aprueba(Juego, robusto) :-
19     not(piezas_pequenias(Juego)).
20 aprueba(Juego, compacto) :-
21     compacto(Juego).
22 aprueba(Juego, no_fragil) :-
23     not(partes_rompibles(Juego)),
24     not(equilibrio_precario(Juego).
25
26 aprobado(Juego, Categoria) :-
27     not(
28         not(aprueba(Juego, Requisito,
29         requisito(Categoria, Requisito)
30     ).
```

Dibuje un círculo alrededor de las reglas y hechos que es necesario introducir en la base de conocimiento para que sea cierto que `aprobado(ludo,familiar).`

a) **[2 pt.]** Es necesario introducir ambos hechos `magnetico(ludo).` y `encastrable(ludo).`

b) **[2 pt.]** Es necesario introducir ambos hechos `simple(ludo).` y `violento(ludo).`

c) **[2 pt.]** Es necesario introducir `simple(ludo).`

d) **[2 pt.]** Si introducimos `jugadores(ludo,2).` será falso que `aprobado(ludo,familiar).`

2. El siguiente script en bash que toma un directorio, crea 10 subdirectorios, lee los archivos del directorio raíz, cuenta las palabras de cada uno y distribuye los archivos en los 10 subdirectorios de forma que el total de palabras por subdirectorio quede lo más balanceada posible.

```
1  ROOT="$1"
2
3  for i in $(seq -w 1 10); do mkdir -p "$ROOT/dir_$i" done
4
5  for file in "$ROOT"/*; do
6      if [ -f "$file" ]; then
7          wc=$(wc -w < "$file")
8          file_words["$file"]=$wc
9          files+=("$file")
10     fi
11 done
12
13 sorted_files=($(for f in "${files[@]}"; do
14     echo "${file_words[$f]}:$f"
15 done | sort -nr | cut -d: -f2))
16
17 for i in $(seq -w 1 10); do dir_wordcount["dir_$i"]=0 done
18
19 for f in "${sorted_files[@]}"; do
20     wc=${file_words[$f]}
21     min_dir=""
22     min_val=999999999
23     for i in $(seq -w 1 10); do
24         dir="dir_$i"
25         total=${dir_wordcount[$dir]}
26         if [ "$total" -lt "$min_val" ]; then
27             min_val=$total
28             min_dir=$dir
29         fi
30     done
31
32     mv "$f" "$ROOT/$min_dir/"
33     dir_wordcount[$min_dir]=$(( ${dir_wordcount[$min_dir]} + wc ))
34 done
```

Indique 2 (dos) números de línea en los que observa las siguientes propiedades de los lenguajes de *scripting*, o ninguno si la propiedad no se encuentra en el siguiente fragmento de código:

a) **[2 pt.]** no declaración de tipos de variables _____

b) **[2 pt.]** abstracciones lingüísticas propias del dominio _____

c) **[2 pt.]** abstracciones lingüísticas para la comunicación con el sistema operativo _____

d) **[2 pt.]** expresiones regulares _____

3. Los siguientes dos fragmentos de código han sido generados por asistentes de programación basados en modelos de lenguaje, pero uno es más inseguro que el otro, porque prioriza la simplicidad del código por encima de su seguridad.

```
1  VERSION A
2  def delete_file(filename):
3      subprocess.run(["rm", "--", filename], check=True)
```

```
1  VERSION B
2  def delete_file(filename):
3      os.system("rm " + filename)
```

[10 pt.] Explique en el espacio provisto aquí (NO USE HOJAS ADICIONALES) cuál es más seguro que el otro y por qué.

4. Dado el siguiente fragmento de código:

```
1  def obtener_entero_positivo(mensaje="Ingresa un entero positivo:"):
2      while True:
3          entrada_usuario = input(mensaje)
4          if not entrada_usuario.isdigit():
5              print("Ingresa un entero positivo:")
6              continue
7          valor = int(entrada_usuario)
8          if valor > 0:
9              return valor
10
11 edad = obtener_entero_positivo("Ingresa tu edad: ")
```

[5 pt.] Nombre o describa muy brevemente la estrategia que se aplica en la línea 7.

Respuesta: _____

[5 pt.] ¿Qué líneas habría que eliminar para que este código no implemente programación defensiva, sino ofensiva?

Respuesta: _____

5. Según el siguiente texto:

> *A synchronized block in Java is synchronized on some object. All synchronized blocks synchronize on the same object and can only have one thread executed inside them at a time. All other threads attempting to enter the synchronized block are blocked until the thread inside the synchronized block exits the block.*

Seleccione las respuestas verdaderas:

a) **[2 pt.]** un bloque sincronizado preserva la atomicidad del bloque a nivel semántico.

b) **[2 pt.]** en un bloque sincronizado se pueden dar condiciones de carrera.

c) **[2 pt.]** la semántica de un bloque sincronizado se puede escribir mediante *locks*.

d) **[2 pt.]** la semántica de un bloque sincronizado se puede escribir mediante un buffer productor-consumidor.

6. **[10 pt.]** El siguiente fragmento de código en Erlang modela la API pública de una red de dispositivos móviles y el bucle de un dispositivo.

```
1  dispositivo(Nombre, Mailbox) ->
2      receive
3          {de, De, Mensaje} ->
4              io:format("[~p] Mensaje de ~p: ~p~n", [Nombre, De, Mensaje]),
5              dispositivo(Nombre, [{De, Mensaje} | Mailbox]);
6
7          {get_mailbox, From} ->
8              From ! {mailbox, Nombre, lists:reverse(Mailbox)},
9              dispositivo(Nombre, Mailbox)
10     end.
11
12 registrar_dispositivo(RedPid, Nombre) ->
13     RedPid ! {registrar, Nombre, self()},
14     receive
15         {ok, registrado, Nombre, Pid} -> {Nombre, Pid};
16         {error, ya_registrado} -> {error, ya_registrado}
17     end.
18
19 enviar_mensaje(RedPid, De, Para, Mensaje) ->
20     RedPid ! {enviar, De, Para, Mensaje}.
21
22 consultar_mailbox(RedPid, Nombre) ->
23     RedPid ! {consultar_mailbox, Nombre, self()},
24     receive
25         {mailbox, Nombre, Mensajes} -> Mensajes;
26         {error, no_encontrado} -> error
27     end.
```

Identifique la(s) línea(s) en las que se aplican las siguentes propiedades del paradigma de actores. Si alguna propiedad no está explícitamente ejemplificada en el texto programa, dejen la línea en blanco:

a) **[2 pt.]** Elasticidad: capacidad para ajustar el número de actores a las necesidades puntuales del sistema en un momento dado _____

b) **[2 pt.]** Robustez en la recuperación de situaciones excepcionales _____

c) **[2 pt.]** Asincronicidad en el manejo de mensajes _____

7. **[10 pt.]** En el espacio que queda (NO USE HOJAS ADICIONALES), provea por lo menos dos razones por las cuals los asistentes de programación basados en modelos de lenguaje suelen tener un mejor desempeño para generar código basado en frameworks, en comparación con código situado en un programa sin inversión de control.