Evaluación

Duración: 3:00 hs.

Ejercicio 1 [evaluación individual del laboratorio]
Responda en pocas líneas las siguientes preguntas en base a la implementación del obligatorio realizada por su grupo:

1. ¿Qué predicados de SWI usaron para la entrada y salida?
2. Enumere las posiciones del tablero, indicando cuántas hay de cada tipo.
3. ¿Qué sucede en su juego si un jugador finaliza el soplido quedando movimientos obligatorios por realizar?

Ejercicio 2 [20 puntos]
Sea el siguiente programa Prolog:

```
p(a).
p(f(f(X)))          :- p(X).

r([],N,[])          :- p(N).
r([X|Xs],f(N),[X|Ys]) :- r(Xs,N,Ys).
r([_|Xs],f(N), Ys)  :- r(Xs, f(N),Ys).
```

y la consulta:

```
? r([1,2,3], f(f(a)), Ys).
```

a) Dé los valores de Ys que son solución de la consulta. Justifique.
b) Modifique el predicado r/3 de forma tal que, agregando uno y solamente un cut:

   i. se mantengan las soluciones
   ii. no tenga soluciones
   iii. tenga una única solución

c) Si se agrega un cut a la primera cláusula de p/1, ¿se modifica la respuesta a la consulta anterior? En caso afirmativo, muestre cuál es. En caso negativo, modifique los argumentos de r en la consulta para que el resultado cambie.

Ejercicio 3 [25 puntos]
Sea el siguiente programa lógico P:

```
p(a, a).
p(f(a), b).
p(f(a), c).
p(g(Y),X)   ← q(X,Y).
q(X, f(X)).
q(X, b)     ← p(X, X).
q(X, c)     ← q(X, Z), p(g(Z), c).
```

y el objetivo G:←`q(a,A), p(f(B),A)`. El intérprete selecciona el átomo de más a la derecha como regla de computación.

a) Dé una interpretación con dominio N, que no sea modelo de P.
b) Defina respuesta computada y respuesta correcta.
c) Construya el árbol SLD correspondiente a $P \cup \{G\}$ con al menos 6 niveles.
d) ¿Existe alguna respuesta correcta que no sea computada? Justifique.
e) Dé las soluciones que muestra el intérprete, si éste selecciona las cláusulas en el orden de aparición en el programa y realiza una búsqueda DFS.

Ejercicio 4 [20 puntos]

a) Implemente el siguiente metaintérprete para *Prolog* puro:

resLim(+G, +N) ← resuelve *G* garantizando que el número de átomos en cualquier subobjetivo no supera los N literales.

El metaintérprete deberá ser capaz de interpretar programas escritos con la sintaxis estándar de Prolog.

b) Construya una gramática DCG que **genere** a*b*. Toda tira del lenguaje debe ser generada. Tenga cuidado con soluciones injustas con a*.

Ejercicio 5 [20 puntos]

Se busca implementar una variante del juego MasterMind. En el mismo participan dos jugadores: el maestro y el hacker. El objetivo del maestro es proponer una clave, que mantiene en secreto, y el del hacker es averiguarla.

El juego se desarrolla de la siguiente manera:

Inicialmente, se genera de manera aleatoria la clave secreta. La misma consiste en una secuencia de cuatro símbolos, posiblemente repetidos, tomados de un alfabeto determinado. Un símbolo puede ser cualquier átomo, siempre y cuando cumpla el predicado `simbolo` (ej.: `simbolo(3), simbolo(rojo)`).

Para encontrar la clave secreta, el hacker prueba combinaciones. En esta versión del juego el hacker puede probar todas las combinaciones que desee. Luego de cada prueba, el maestro indica cuántos símbolos de la prueba coinciden en lugar y forma con símbolos de la clave. Por ejemplo, si la clave es `[verde, rojo, azul, blanco]` y la prueba es `[negro, rojo, azul, verde]`, el maestro debe informar que existen dos coincidencias.

En esta variante la computadora asumirá el rol del hacker y el humano el del maestro. Para resolver el problema se pide implementar los siguientes predicados:

`alfabeto(-Alfa)` ← **`Alfa`** es una lista que representa al conjunto de todos los términos que cumplen el predicado **`simbolo`**.

`clave_maestro(-Clave)` ← **`Clave`** es una lista que contiene los cuatro símbolos que componen la clave, tomados al azar a partir del alfabeto. Los símbolos de la clave pueden estar repetidos.

`coincidencias(+Prueba, -N)` ← Le pregunta al usuario cuántos símbolos de **`Prueba`** coinciden con los símbolos de la clave y lee de la entrada estándar la respuesta, que se instancia en **`N`**. La respuesta **`N`** debe ser un número entre 0 y 4. Mientras esto no ocurra, deberá desplegar un mensaje de error y volver a realizar la pregunta.
Nota: Se debe controlar que el stack no crezca.

Suponiendo que están definidos los siguientes símbolos:

```
simbolo(verde). simbolo(amarillo). simbolo(azul). simbolo(rojo).
```

Ejemplos de consultas:

Consulta	**Resultado**
`?alfabeto(Alfa)`	`Alfa = [verde, amarillo, azul, rojo]`
`?clave_maestro(Clave)`	`Clave = [rojo, amarillo, verde, rojo]`

`?coincidencias([verde, amarillo, azul, azul], N).`	`La prueba es: [verde, amarillo, azul, azul]`
	`Cuantas coincidencias hubo? 5.`
	`Error!`
	
	`La prueba es: [verde, amarillo, azul, azul]`
	`Cuantas coincidencias hubo? 1.`
	
	`N = 1`