

## PLP - Segundo Parcial - 1<sup>er</sup> cuatrimestre de 2022

Este examen se aprueba obteniendo al menos dos ejercicios bien menos (B-) y uno regular (R). Las notas para cada ejercicio son: -, I, R, B-, B. Entregar cada ejercicio en hojas separadas. Poner nombre, apellido, número de orden y cantidad de hojas en la primera hoja, y numerar las hojas. Se puede utilizar todo lo definido en las prácticas y todo lo que se dio en clase, colocando referencias claras.

El orden de los ejercicios es arbitrario. Recomendamos leer el parcial completo antes de empezar a resolver.

### Ejercicio 1 - Programación Lógica

Implementar los predicados respetando en cada caso la instanciaión pedida. Los generadores deben cubrir todas las instancias válidas de aquello que generan sin repetir dos veces la misma. No usar cut (!) ni predicados de alto orden como `setof`, con la única excepción de `not`.

- a) Definir el predicado `masRepetido(+L,-X)` que es verdadero cuando X es el elemento con el mayor número de apariciones dentro de la lista L. Notar que puede haber más de un elemento que cumpla con esta condición, por lo que se deberá instanciar a X con todas las soluciones.
- b) Definir el predicado `partesQueSuman(+L,+S,-P)` que es verdadero cuando P es una lista con elementos de L que suman S. Por ejemplo:

```
?- partesQueSuman([1,2,3,4,5],9,P).
P = [1, 3, 5] ;
P = [2, 3, 4] ;
P = [4, 5] ;
false.
```

### Ejercicio 2 - Resolución

- a) Representar en forma clausal las siguientes fórmulas de lógica de primer orden referidas a números enteros.
  - i.  $\forall x (par(x) \supset \exists y (y > x \wedge \neg par(y)))$  - Para todo  $x$  par existe un impar mayor que él.
  - ii.  $\forall x (\neg par(x) \supset \exists y (y > x \wedge par(y)))$  - Para todo  $x$  impar existe un par mayor que él.
  - iii.  $\forall x \forall y \forall z ((x > y \wedge y > z) \supset x > z)$  - La relación de mayor es transitiva.
- b) Usando resolución demostrar que para todo par existe otro par mayor, es decir,  
 $\forall x (par(x) \supset \exists y (y > x \wedge par(y)))$ .
- c) Indicar si la demostración es SLD y justificar.

### Ejercicio 3 - Objetos

- a) Definir en Cálculo Sigma la clase Pila, cuyas instancias puedan realizar las siguientes operaciones:
  - `isEmpty`: un booleano que indica si la pila se encuentra vacía.
  - `push`: función que dado un elemento  $x$  agrega  $x$  a la pila, devolviendo la nueva pila.
  - `top`: devuelve el último elemento agregado a la pila, sin retirarlo de la pila.
  - `pop`: elimina el último elemento agregado a la pila, devolviendo la nueva pila.

El método `new` debe devolver una pila vacía. Cuando la pila está vacía, `pop` deberá devolver la misma pila, y `top` debe colgarse.

- b) Definir en JavaScript el objeto `emptyStack`, cuyo comportamiento sea análogo al de la pila vacía del punto a), pero que no tenga definido el mensaje `top`, sino que este se defina únicamente para las pilas no vacías.

1	2	3
B--	B-	B-

Solución

1) % msn Repetido (+L, -X)

, nunca en más Right.

 $\text{msnRep}(L, X) \equiv \text{append}(\text{Left}, [X | \text{Right}]), \text{msn}(\text{menab}, (X, \text{LEFT}))$ , $\text{contibd}(X, L, N), M \in N+1, \text{msn}((\text{menab}(Y, L), \text{contibd}(Y, L, M)))$ .Esto solo dice que no hay otro elemento que aparezca  $N+1$  veces, pero puede haber uno que aparezca  $N+2$  veces o más.

% contibd (+E, +L, -N)

 $\text{contibd}(-, [], 0)$ . ✓ $\text{contibd}(H, [H | T], N) \equiv \text{contibd}(H, T, M), N \in M+1$ . ✓ $\text{contibd}(E, [H | T], N) \equiv E = H, \text{contibd}(E, T, N)$ . ✓

Idea: Iterar una vez toda la lista de los elementos de la lista, así no arreglamos de no ver más de una vez el mismo elemento. Despues chequemos que tarda tiempo ~~más~~ <sup>mín</sup> más no es necesario  $N+1$ .

2) % poner Que Suman (+L, +S, -P) -- genera ST tail

 $\text{ponerQueSuma}(L, S, P) \equiv \text{poner}(L, \text{ListaDePoner}), \text{menor}(\text{UnaParte}, \text{ListaDePoner}),$   
 $\text{menor}(\text{UnaParte}, S)$ . ✓

% menor (+L, ?S) (dónde con S instanciada).

 $\text{menor}([], 0)$ . $\text{menor}([H | T], N) \equiv \text{menor}(T, ST), N \in ST + H$ . ✓

% poner (+L, -ListaDePoner)

 $\text{poner}([], [L])$ . $\text{poner}([H | T], \text{ListaDePoner}) \equiv \text{poner}(T, \text{ListaDePonerTail}), \text{agregarATodos}(\text{ListaDePonerTail}, H, P),$   
 $\text{append}(\text{ListaDePonerTail}, P, \text{ListaDePoner})$ .

% agregonATodos(+LL,+E,-LLOut) :- Agrega E al final de cada lista de LL.

agregarATodos ([[ ]], E, [[ E ]]). Estos casos no son disjuntos ~~se~~ y falta el caso de [ ].

agregarATodos ([ H | T ], E, [ H2 | T2 ]) :- <sup>append</sup> concat(H, [ E ], H2), agregarATodos(T, E, T2).

$$\text{II.a. } \forall x [P_m(x) \supset \exists y [y > x \wedge \neg P_m(y)]]$$

$$\forall x [\neg P_m(x) \vee \exists y [y > x \wedge \neg P_m(y)]]$$

$$\forall x \exists y [\neg P_m(x) \vee (y > x \wedge \neg P_m(y))]$$

$$\forall x \exists y [(\neg P_m(x) \vee y > x) \wedge (\neg P_m(x) \vee \neg P_m(y))]$$

$$\forall x [(\neg P_m(x) \vee f(x) > x) \wedge (\neg P_m(x) \vee \neg P_m(f(x)))]$$

$$\left\{ \{\neg P_m(x_1), f(x_1) > x_1\}, \{\neg P_m(x_2), \neg P_m(f(x_2))\} \right\} \checkmark$$

$$\text{II. } \forall x [\neg P_m(x) \supset \exists y [y > x \wedge P_m(y)]]$$

$$\forall x [P_m(x) \vee \exists y [y > x \wedge P_m(y)]]$$

$$\forall x \exists y [P_m(x) \vee (y > x \wedge P_m(y))]$$

$$\forall x \exists y [(P_m(x) \vee y > x) \wedge (P_m(x) \vee P_m(y))]$$

$$\forall x [(P_m(x) \vee f(x) > x) \wedge (P_m(x) \vee P_m(f(x)))] \times$$

Grante la misma función de Shéla para dos variables distintas (esta y la de f).

$$\left\{ \{P_m(x_3), f(x_3) > x_3\}, \{P_m(x_4), P_m(f(x_4))\} \right\}$$

$$\text{III. } \forall x \forall y \forall z [(x > y \wedge y > z) \supset x > z]$$

$$\forall x \forall y \forall z [\neg(x > y \wedge y > z) \vee (x > z)]$$

$$\forall x \forall y \forall z [\neg x > y \vee \neg y > z \vee x > z]$$

$$\left\{ \{\neg x_s > y_s, \neg y_s > z_s, x_s > z_s\} \right\} \checkmark$$

### Segunda parte de la hoja

$$1 \{ \delta(x_1) > x_1, \neg P_m(x_1) \}$$

$$2 \{ \neg P_m(x_2), \neg P_m(\delta(x_2)) \}$$

$$3 \{ \delta(x_3) > x_3, P_m(x_3) \}$$

$$4 \{ P_m(x_4), P_m(\delta(x_4)) \}$$

$$5 \{ x_5 > y_5, \neg y_5 > z_5, x_5 > z_5 \}$$

$$6 \{ P_m(a) \}$$

$$7 \{ \neg y_6 > a, \neg P_m(y_6) \}$$

$$1,6 \quad x_1 > a : P_m(a) \text{ con } \neg P_m(a) ?$$

$$8 \{ \delta(a) > a \} \checkmark$$

$$2,6 \quad x_2 > a : P_m(a) \text{ con } \neg P_m(a)$$

$$9 \{ \neg P_m(\delta(a)) \} \checkmark$$

$$9,3 \quad x_3 > \delta(a) : P_m(\delta(a)) \text{ con } \neg P_m(\delta(a))$$

$$10 \{ \delta(\delta(a)) > \delta(a) \}$$

$$9,4 \quad x_4 > \{ a \} : P_m(\{ a \}) \text{ con } \neg P_m(\{ a \})$$

$$11 \{ P_m(\{ \delta(\delta(a)) \}) \}$$

$$5,8 \quad y_5 > \delta(a) : \text{ bajo } \neg P_m(a)$$

$$12 \{ \neg x_5 > \{ a \}, x_5 > a \} \checkmark$$

$$10,12 \quad x_5 < \{ a \} : \{ \{ a \} \} > \{ a \} \text{ con } \neg \{ \{ a \} \} \Delta \{ a \}$$

### Tercera parte de la hoja

$$13 \{ f(f(a)) > a \} \checkmark$$

$$7,13 \quad y_6 < \{ f(a) \} : f(f(a)) > a \text{ con } \neg f(f(a)) > a$$

$$14 \{ \neg P_m(f(f(a))) \} \checkmark$$

$$11,14 \quad P_m(f(f(a))) \text{ con } \neg P_m(f(f(a)))$$

$$15 \{ \square \} \checkmark$$

Plan: Primera bencima a un

último mayor que a, sea de

$f(a)$ . Despus bencima a un

por mayor que  $f(a)$ , sea  $\square = f(f(a))$ .

Por ultimo, un contradiccion argumento.

que  $f(f(a)) > a$ .

c) No se SLD porque no bencima

la demostración de forma lineal.  $\otimes$

Ej: Para conseguir  $\neg \{ \dots \}$  no

usarán  $\neg \neg$ .

$\otimes$  (Entre otras cosas)

b) Empeñarnos por negar lo que queremos demostrar: Cuarta parte de la hoja

$$\neg (\forall x [P_m(x) \supset \exists y [y > x \wedge P_m(y)]])$$

$$\neg (\forall x [\neg P_m(x) \vee \exists y [y > x \wedge P_m(y)]])$$

$$\neg (\forall x \exists y [\neg P_m(x) \vee (y > x \wedge P_m(y))])$$

$$\exists x \forall y [\neg P_m(x) \wedge \neg (y > x \wedge P_m(y))]$$

$$\exists x \forall y [P_m(x) \wedge (\neg y > x \vee \neg P_m(y))]$$

$$\forall y [P_m(a) \wedge (\neg y > a \vee \neg P_m(y))]$$

$$\{\{P_m(a)\}, \{\neg y_6 > a, \neg P_m(y_6)\}\} \checkmark$$

A partir de acá corrijo como si todos los chicos/as fueran correctos.

3) a) Pila.  $\text{def } [\text{new} = \lambda(z)[\text{isEmpty} = \lambda(s)z.\text{isEmpty}(s), \text{push} = \lambda(s)\lambda(e)z.\text{push}(s)(e)],$   
 $\text{top} = \lambda(s)z.\text{top}(s), \text{pop} = \lambda(s)z.\text{pop}(s)]$

$\text{isEmpty} = \lambda(s) \text{ true}$

$\text{push} = \lambda(s)\lambda(e)((s.\text{isEmpty} := \text{false}), \text{top} := e), \text{pop} := s$

$\text{top} = \lambda(s) \text{ Pila}. \text{top}(s)$  X Estás usando Pila dentro de la definición de Pila.

Los macros no son recursivos.

$\text{pop} = \lambda(s) s$  Lo correcto sería escribir  $\lambda(s) s.\text{top}$ .

I

b) let emptyStack = {

$\text{isEmpty} = () \Rightarrow \{\text{return true}\},$  ¿Por qué no true directamente?

$\text{pop} = \text{function}() \{\text{return thin}\}$

$\text{push} = \text{function}(e) \{$

$\text{return }$

$\text{isEmpty} = () \Rightarrow \{\text{return false}\},$

$\text{top} = () \Rightarrow \{\text{return e}\},$

$\text{pop} = () \Rightarrow \{\text{return thin}\},$

$\text{push} = \text{thin.push}$  ¿Por qué todos los atributos son funciones?

I

Debería hacer  $\text{emptyStack.push}(e)$ , this no va ser emptyStack.

Por lo tanto, si haces por ejemplo  $\text{emptyStack.push}(1).\text{push}(2)$ , el segundo push va a ser al push de emptyStack.

I

No está bueno que las pilas y sus subpilas no tengan un prototipo en común, ya que eso impide implementar funcionalidades comunes a todas.