CC41A Lenguajes de Programación
Examen - Otoño 2006
Profesor: Luis Mateu

Pregunta 1

La relación `sumaParcial(NumL, Sum)` indica en Prolog que `NumL` es una lista de números y que existe un subconjunto de los números contenidos en ella que sumados dan el número `Sum`. Ejemplos:

- `?- sumaParcial([7, 2, 8, 5], 15)` => Si, porque 7+8=15
- `?- sumaParcial([7, 2, 8, 5], 4)` => No
- `?- sumaParcial([7, 2, 8, 5], 22)` => Sí, porque 7 + 2 + 8 +5= 22
- `?- sumaParcial([7, 2, 8, 5], 8)` => Sí, porque 8 = 8.
- `?- sumaParcial([7, 2, 8, 5], 20)` => Sí, porque 7 + 8 + 5 = 20.
- `?- sumaParcial([7, 2, 8, 5], 0)` => Sí, la suma de elementos en el subconjunto vacío es 0.

Implemente en Prolog la relación `sumaParcial`. *Hint*: descomponga el problema en (i) una relación que indique si una lista S es una sublista de otra lista L, y (ii) otra relación para indicar que la suma de los elementos de la lista L es Sum.

Pregunta 2

En una aplicación se necesita trabajar con expresiones booleanas en *Smalltalk*. Mediante el siguiente *ejemplo de uso*, se ha especificado informalmente el comportamiento de las expresiones booleanas. Observe que las clases y métodos que Ud. debe considerar aparecen en **negritas**.

Evaluación en el Transcript	*Nota*	*Resultado en pantalla*

`|t f a aa n e|`
`t := BVerdadero new.`
`f := BFalso new.`
`a := t y: f.`
`n := f negado.`
`e := n y: (a not).`
`e mostrar.`
	A
	B
	C
	D
	
	E
		`((~F)^(~(V^F)))`

`Transcript show: (f prof)`
`Transcript show: (n prof)`
`Transcript show: (e prof)`
	F	1
		2
		3

`aa := a y: f.`
`Transcript show: aa.`
`Transcript show: (aa prof)`
		`((V^F)^F)`
		3

Entonces se requiere que una expresión booleana pueda ser (i) la expresión que representa verdadero, como en A, (ii) la expresión que representa falso, como en B, (iii) la conjunción de dos expresiones, como en C, y (iv) la negación de una expresión, como en D. Además se requiere un método que muestre en pantalla la expresión sin evaluar (ver E) y un método que determine la profundidad del árbol de evaluación de la expresión (ver F).

En concreto se pide que implemente una jerarquía de clases en Smalltalk que funcione de acuerdo al ejemplo de uso dado. Se requiere que Ud. utilice un enfoque *orientado a objetos* en su diseño. Para cada clase indique (a) la clase base, (b) las variables de instancia, y (c) los métodos de instancia.

Observaciones: Para (a) y (b) *no necesita* utilizar la sintaxis rigurosa de Smalltalk. Para (c) sí debe usar la sintaxis de implementación de métodos de Smalltalk. En su diseño seguramente requerirá agregar métodos y clases que no aparecen en el ejemplo de uso.

Pregunta 3

a) Suponga que Ud. ejecuta el mismo programa en (i) Scheme, (ii) un dialecto de Scheme que pasa los parámetros por valor/resultado, (iii) un dialecto de Scheme que pasa los parámetros por referencia. Discuta cómo podría ser afectado el resultado del programa por el paso de parámetros utilizado por el lenguaje, cuando el programa ejecutado es *estrictamente funcional*.

b) Confeccione una tabla que incluya en las filas los lenguajes Java, C, Scheme, Smalltalk y Prolog. En las columnas indique para cada lenguaje (i) si es dinámicamente tipado, (ii) si es robusto, (iii) un índice de eficiencia de ejecución, con un 1 para el más ineficiente, y 10 para el más eficiente, (iv) si pueden ocurrir referencias colgantes (*dangling references*), y (v) la frase más adecuada: *backtracking*, orientación a objetos, ausencia de efectos de borde, compilador just-in-time, aritmética de punteros.

c) Escriba una especificación para CUP que reconozca si la entrada es una secuencia de abre y cierre de paréntesis bien balanceado y despliegue además en pantalla la máxima profundidad alcanzada en los paréntesis. Ejemplos:

Entrada	***Resultado***
)(	rechazado 0
( )	1
(( )(( )))	3