Final de Paradigmas de Programación 1/12/2012 Nombre:

Ejercicio 1

Dada la siguiente solucion en Prolog:

```
tiene( juan, auto).
tiene( ana, moto).
tiene( juan, casa).
...
cantidadDeCosas(Persona, Cantidad):-
        findall(  Cosa, tiene( Persona, Cosa), Cosas),
        length(Cosas, Cantidad).
```

1. Analizar la inversibilidad del predicado cantidadDeCosas, mostrando los diferentes modos de consultas posibles con sus correspondiente respuestas.
2. ¿Hay alguna mejora que se le pueda implementar? ¿Cuál? Justificar.
3. Hacer el equivalente de la solución mejorada en el paradigma funcional. ¿Cómo se suple la ausencia de inversibilidad?

Ejercicio 2

El zoológico de Berlín trae animales de todas partes del mundo. Su principal requerimiento es **saber si todos los animales se adaptan bien**. Se sabe que hay monos, jirafas y leones, donde:

- Los monos están bien adaptados si tiene comida nutritiva.
- Las jirafas están bien adaptadas si pesan más de 500 kilos y tienen comida balanceada.
- Los leones están bien adaptados si tienen comida natural.

(existen métodos ya hechos que informan cómo es la comida)

1. Plantee dos soluciones para este requerimiento, una utilizando polimorfismo y otra sin él. No es necesario codificar completamente las propuestas, incluya únicamente el código indispensable para explicarlas. Compare ambas soluciones indicando fortalezas y debilidades de cada una.
2. Ahora el zoológico quiere contemplar a otros animales que se adaptan bien si pesan más de 500 kilos, tienen comida balanceada y fueron vacunados. Codificar la solución y justificar. ¿Qué concepto evita la duplicación de código?
3. Volviendo al requerimiento original desarrolle la solución en el paradigma lógico utilizando adecuadamente polimorfismo y orden superior. Justificar las fortalezas en la utilización de los conceptos mencionados.

Ejercicio 3

Dadas las siguientes definiciones

```
metodoLoco
```

```
	^self copyFrom: 1 to: (self size // 2)

funcionLoca algo = take (length algo `div` 2) algo

predicadoLoco(L1, L2):- length(L1, C), C2 is C//2, takelist(L1,C2,L2).*
```

1. ¿Qué hacen?
2. ¿Con qué tipos de datos operan?
    a. En Smalltalk, indicar en qué clase podría estar definido el método.
    b. En Haskell, definir el tipo de la función.
    c. En Prolog, explicar con palabras los tipos de datos de los argumentos del predicado.
3. Mostrar al menos dos ejemplos de invocación y respuesta en cada caso, donde se vea la amplitud de tipos de datos que soportan.
4. En base a las definiciones dadas y lo indicado en los dos puntos anteriores compare los conceptos de polimorfismo en los tres paradigmas. ¿En qué lugar identifica su uso? ¿Qué es lo “polimórfico”? ¿Qué ventajas trae aparejadas?

**takelist es un predicado que relaciona una lista, un numero n y otra lista que tiene los n primeros elementos de la primer lista.*