Programación Declarativa Aplicada – Curso 2016-17 - Práctica 1

Evaluación: Entrega en clase: 0.25. Entrega la siguiente semana: 0.15 (más adelante no puntúa)

1) Bertoldo, estudiante de Erlang, ha tecleado la siguiente expresión complicada en la consola de Erlang:

```
f().
B=[C=6 | D=9], E = tl(B), F=hd(E).
```

Sin embargo, obtiene un error que no entiende. Indica la razón del error.

2) La funcion `io:read(msg)` escribe en consola el mensaje msg y después espera que se escriba un termino Erlang T (seguido de punto.). Si no hay error, el valor devuelto es {ok,T}.
Utiiza esta función para escribir una expresión que pida dos números y devuelva la suma.

3) Supongamos que la variable S está ligada a un string. Escribir una expresión que indique si el string S es palíndromo (es igual al string que se obtiene al darle la vuelta). La expresión devolverá true si en efecto es palíndromo o false en caso contrario. Ayuda: usar la BIF lists:reverse

4) Una interesante posibilidad de Erlang es la de asignar funciones a variables. Por ejemplo

```
f(), A=fun(X) -> X+1 end.
```

liga A a una función que dado un valor X devuelve X+1. Ahora podemos hacer :

```
A(3).
4
A(A(4)).
6
```

Se pide: ligar una variable con nombre Rev a una función que recibe una lista L como argumento e indica si la lista es un palíndromo.
Los siguientes ejemplos confirmarán que hemos definido bien la función:

```
Rev([a,bb,c,bb]).
false
Rev([3,bb,bb,3]).
true
Rev("dabalearrozalazorraelabad").
true
```

En el resto de la práctica vamos a trabajar con figuras geométricas representadas como:

- Cuadrado con lados paralelos a los ejes: {square,{x,y}, l}, con x,y la coordenada superior izquierda y l la longitud del cuadrado. Por ejemplo el cuadrado de lado 20 y situado en la coordenada (5,2) se escribe {square,{5,2}, 20}
- Rectángulo con lados paralelos a los ejes: {rectangle,{x,y}, lx, ly}, (x,y) esquina superior izquierda, lx longitud del lado en el eje x, longitud del lado en el eje y.
- Círculo: {circle,{x,y},r}, con {x,y} el centro y r el radio del círculo.

5) Supongamos que la variable C contiene un cuadrado (no hace falta comprobarlo, es seguro que es así). Escribir una expresión que muestre el área de C.

6) `lists:keysearch(Key, N, TupleList) -> {value, Tuple} | false`
es una función de la biblioteca lists de Erlang que busca en la lista de tuplas TupleList la primera

tupla que verifique que su enésimo (N) componente es el valor Key. Si existe esta tupla, devuelve {value,Tuple}, y si no existe devuelve false.
Supongamos que L es una lista de figuras geométricas. Escribir una expresión que, apoyándose en esta función, calcule el área del primer cuadrado contenido en la lista.

7) Supongamos que la variable R contiene un rectángulo. Escribir una expresión que compruebe si se trata de un cuadrado (lados iguales). Si es un cuadrado la expresión se evaluará (el valor de salida será) un cuadrado equivalente, en otro caso se obtendrá un error de matching.

8) Suponemos que tenemos dos variables S1, y S2 que representan dos figuras. Escribir una expresión Erlang que devuelva true si S1 es círculo, S2 es un rectángulo y S1 está incluido en S2. En otro caso la expresión fallará (error de matching si S1 no es un círculo o S2 no es un rectángulo) o devolverá `false`.

9) En Erlang una función se llama predicado cuando devuelve `true` o `false`. Por ejemplo, la función contenida en la variable `Rev` del ejercicio 4 es un predicado.

Escribir una función que indique si una figura es un cuadrado (su primer elemento es '`square`'), y ligar la variable `Cuadrado` a esta función.
Ayuda: la función `element(N,Tuple)` devuelve el elemento que ocupa la N-posición de una tupla (el primer elemento ocupa la posición 1).

Ejemplos para comprobar si es correcto:

```
Cuadrado({square,{7,8},11}).
true
Cuadrado({rectangle,{7,8},11,14}).
false
Cuadrado({square,tururu}).
true
```

10) Los predicados son muy útiles y se usan como argumentos en muchas funciones. Por ejemplo la función

```
lists:filter(Pred, List1) -> List2
```

toma como entrada un predicado `Pred` y una lista. El valor `List2` corresponde a aquellos elementos de `List1` que verifican el predicado `Pred`. Por ejemplo:

```
Positive = fun(X) - > X>0 end.
lists:filter(Positive,[5,-3,2,6]).
[5,2,6].
```

Otra función interesante es

```
lists:map(Fun, List1) -> List2
```

En este caso `Fun` es cualquier función con un argumento. Lo que hace `map` es aplicar `Fun` a todos los elementos de `List1`. El resultado es una nueva lista `List2`.

Ejemplo:

```
Inc=fun(X) -> X+1 end.
lists:map(Inc,lists:seq(-10,10)).% lists:seq(A,B) produce la lista [A,A+1….B]
```

```
[-9,-8,-7,-6,-5,-4,-3,-2,-1,0,1,2,3,4,5,6,7,8,9,10,11]
```

Ambas se pueden anidar

```
filter(Positive,lists:map(Inc,lists:seq(-10,10))).
[2,3,4,5,6,7,8,9,10,11]
```

Se pide: usando lists::map y lists::filter, ligar la variable Cs a una función que dada una lista de figuras devuelva una lista con las áreas de los cuadrados contenidos en la lista. Si hace falta se puede definir alguna función auxiliar.

Ejemplo para probar:

```
Cs([{square,{4,5},5},{rectangle,{0,0},8,8},{square,{1,2},8}]).
[25,64]
```