Práctica 4

(entrega hasta el 1 de noviembre de 2016)

1. [1] Define (en un fichero pract4.hrl):
   - una tipo registro (record) para representar una carta de una baraja de póker. La carta viene determinada por un *valor* (as,2,3,. . . ,10,j,q,k) . . . y un *palo* (picas,rombos,corazones,tréboles). Por defecto el registro creará el as de corazones.
   - Define asimismo un tipo registro *mano* con un solo campo *cartas* que representa un conjunto de cartas de la baraja mediante una lista de cartas y que toma la lista vacía como valor por defecto.
   - Una estructura de record *persona*, con información sobre nombre y apellidos, documento de identidad, edad, calle, ciudad, y cualquier otra cosa que se te ocurra.
   - Finalmente usa una macro para definir una constante VALORES que tenga el valor [as,2,3,4,5,6,7,8,9,10,j,q,k].

Nota: Podemos probar a descomentar los tres primeros test del fichero pract4_test() simplemente para probar que la sintaxis seguida es la correcta.

El resto de los ejercicios los escribiremos en un fichero **pract4.erl**.

2 [1.5] Programa una función *poker* que devuelva *true* si una mano (estructura del ejercicio anterior) tiene exactamente 4 cartas con el mismo valor y *false* en caso contrario

Nota:
   - No hace falta comprobar el palo de las 4 cartas del mismo valor
   - La mano puede contener otras cartas, el tamaño de la mano no es conocido
   - Para probar la función puedes descomentar los tests pokerSI_test() y pokerNO_test, que comprueban una mano con póker y otra sin póker

3.[1.5] Programa la función *edad(Persona)* que devuelva la edad de un record de tipo persona.

4.[1.5] *vecinos(P1, P2)* = true si P1 y P2 son personas que viven en la misma calle de la misma ciudad, false en otro caso.

5.[1.5] *habitantes(Ps, Ciudad)* = personas de la lista Ps que viven en Ciudad

6.[1.5] incluye(P,Ps) = resultado de incluir la persona P en la lista de personas Ps; si en Ps ya hay una persona P' con el mismo documento de identidad de P se devuelve Ps.

7.[1.5] Si probamos la expresión: `lists:map(fun(X)->1/X end,[1,2,3,4,0,5]).` veremos que se lanza una excepción. Escribe una versión de la función map llamada *mapSafe* que en caso de producirse una excepción incluya el valor *error* en la lista de salida y continue procesando el resto de la lista.

```
Por ejemplo (ver fichero test):
pract4:mapSafe(fun(X)->1/X end,[1,2,3,4,0,5]).
[1.0,0.5,0.3333333333333333,0.25,error,0.2]
```

Nota: Al finalizar la práctica se entregarán dos ficheros, pract4.hrl y pract4.erl