PEC 2 - SISTEMA PREVENCIÓN G.A.S.

JAVIER COLMENERO FERNÁNDEZ

1 - DESCRIPCIÓN DEL CONOCIMIENTO DEL DOMINIO

El tema elegido para la realización de esta práctica ha sido **el consumismo**.
La sociedad actual en la que vivimos se puede decir que es muy consumista, ya que probablemente muchas personas, compran o gastan el dinero en más cosas de las que deberían/necesitan.

Hay un tipo muy peculiar de consumismo que es el llamado GAS.

Se usa el término GAS, para definir a aquellas personas que sienten una fuerte necesidad de adquirir, o seguir adquiriendo objetos de un determinado campo de trabajo o de hobby.
En inglés son las siglas de Gear Acquisition Syndrome (o en español: Síndrome de Adquisición de Equipamiento).
Lo sufren en general guitarristas, (que fué para los que se inventó el término, bajo el acrónimo de Guitar Acquisition Syndrome) aunque se puede extender a cualquier persona con un hobby que involucre objetos. (fotografía, surf).

Así un aficionado a la fotografía podría sentir la necesidad cada dos por 3 de adquirir nuevas cámaras, lentes, trípodes, software de edición de imágenes.

El sistema basado en reglas de esta práctica surge con el ánimo de persuadir al usuario de comprar algo que realmente no necesita.

Para ello se propone un test que el usuario debe responder. Al finalizar el test, y en base a la puntuación asignada a cada respuesta, el sistema será capaz de definir qué nivel de GAS le corresponde al usuario (ninguno, bajo, medio, alto). Estos niveles provienen de mi conocimiento experto sobre el GAS, ya que por propia experiencia sé que puedes no tener ninguno, o bien tener un poco, bastante o mucho.

Una vez hallado este nivel de GAS, el sistema proporcionará una respuesta para persuadir al usuario de su compra.
Hay una serie de consejos diferentes para cada nivel de GAS, y el sistema responderá con uno de ellos de forma aleatoria. No le responderá lo mismo a alguien con un nivel de GAS nulo que a alguien con un GAS alto. Dado que la respuesta es al azar entre varias disponibles, no siempre que se tengas un nivel de GAS bajo se le dará la misma respuesta.
Así se permite añadir nuevas respuestas de persuasión que podrán ser dadas al usuario con un determinado nivel de GAS.

2 - METODOLOGÍA DE DESARROLLO.

Lo primero ha sido la elección del tema (consumismo), y dentro de este tema concretar una finalidad para el sistema basado en reglas. En mi caso decidí implementar un “Sistema Anti GAS” intentando persuadir al usuario con frases más o menos fuertes dependiendo del nivel de GAS que este tenga.

Una vez hecho esto, me puse a comprender los principios de PROLOG, peleandome con las listas o la recursividad como recurso para resolver muchos problemas de la programación lógica.

He echado una ojeada a prácticas de otros años para ver cómo se hace un uso correcto del lenguaje Prolog. Me ayudó especialmente para la parte de hacer un menú interactivo, ya que con el escaso conocimiento que adquirí de las bases de prolog no fuí capaz de sacar un método en programación lógica donde se muestre un menú interactivo.

Con estas ideas tan básicas del dominio, o sea, los niveles de gas (ninguno, bajo, medio alto), y una serie de frases de persuasión para cada nivel, me puse a codificar, intentando hacer un predicado lógico que en base a una puntuación asigne uno de estos niveles.

Mi objetivo inicial era establecer la relación entre los niveles de GAS, las respuestas de cada nivel y cómo seleccionar una de ellas al azar.

Así uno de los primeros predicados que codifiqué fué ***gasGrade(Num, Grade).*** El cual dependiendo de cual sea el valor numérico de *Num*, te devuelve uno de los 4 niveles de GAS.

Después necesitaba que dependiendo del Grado de GAS obtenido pudiera acceder a una de las listas de consejos para cada grado. Se me ocurrió que una manera sencilla de establecer la relación era definiendo un predicado con varias reglas. Así surgió el predicado **persuadeByGrade**, que tienen una definición para cada nivel de GAS. entonces **persuadeByGrade(ninguno)**, contendría las acciones a realizar para una persona sin GAS. De esta manera fui capaz de asignar a cada nivel de gas, definido como constantes, con su lista de consejos de persuasión, definidos con los predicados sin variables**.**

Con esta estructura general de la práctica, me puse con el caso particular de las preguntas. Cuales tenían que ser, que respuestas y que puntuaciones asignar a cada respuesta.

Decidí un método sencillo, implementando con una lista de 4 elementos **(questionPoints**) los puntos a cada una de las posibles respuestas. Así si el usuario elige la segunda opción, se añadía a la lista de puntos que va guardando la puntuación (**pointList)** aquella que estuviera en segunda posición el la lista **questionPoints.**
Así por último tuve que afinar qué puntuación asignar a cada posible respuesta, y en base al total de puntos obtenidos retocar el predicado **gasGrade** para que devolviera lo mejor posible el grado de GAS del usuario.

3 - DESCRIPCIÓN DE LA ESTRUCTURA DE LA BASE DE REGLAS Y CONSIDERACIONES DE EFICIENCIA.

En la base de conocimientos podemos encontrar:

- Constantes
    - Los niveles de GAS: ninguno, bajo, medio, alto

- Hechos sin variable
    - Serían los listados de consejos en base al nivel de GAS, los predicados **gasAnswer**, contienen un dos términos, el primero es el nivel de GAS definido como constante y el segundo una lista con las frases de persuasión para ese nivel de GAS

- Hechos con variables dinámicas, como el predicado **pointList**([]), el cual se encarga de ir almacenando las respuestas a cada una de las preguntas del test
- Predicados para el inicio de la aplicación, la impresión del menú y las acciones a llevar a cabo según la opción de menú elegida.
- Predicados que representan las preguntas a mostrar al usuario.
- Predicados auxiliares para limpiar la pantalla, manipular la lista de puntos, sumar la puntuación total, elegir un consejo de una lista, etc…

Como consideraciones de eficiencia, me plantée el uso de las listas para aglutinar las respuestas a cada nivel de GAS. Para establecer la relación entre el nivel de gas asignado y las respuestas a cada nivel creé en un principio 4 predicados, gasNone, gasLow, gasMid, gasHight y cada uno de ellos se usaba en el predicado **persuadeByGrade,** y tenía definido 4 reglas para ese predicado **persuadeByGrade(niguno). persuadeByGrade(bajo), persuadeByGrade(medio), persuadeByGrade(alto).** En cada uno de estos predicados usaba en el antecedente la llamada a **gasNone** si era ninguno la constante, **gasLow** con bajo… y así sucesivamente

Después me percaté de que podía hacer un predicado genérico con una sola regla, usando el Grado de gas como argumento con lo que al final quedó asi:

```
% Hechos sin variable
% Este predicado relaciona los niveles de GAS con las respuestas a
dar
gasAnswer(ninguno, [
    'No tienes GAS. Eres un tio con cabeza. Seguro que sabes cuando
darte un capricho y cuando no.',
    'No tienes GAS. Prueba a comprarte algo hombre. Tu no tienes
peligro de caer en el GAS.',
    'No tienes GAS. Enhorabuena. Si quieres comprar algo puedes
hacerlo']).

gasAnswer(bajo, [
    'Tienes un poco de GAS. No es preocupante, mucha gente le
entusiasma su hobby, mientras no vaya a mas no hay problema.',
    'Apenas tienes GAS, no hay problema mientras no vayas a mas.',
    'No te preocupes no tienes casi GAS, sigue pensando que compras
te aportan algo y cuales no.']).
gasAnswer(medio, [
    'Tu nivel de GAS es preocupante. De aquí a despilfarrar el dinero
hay un paso. Piensa bien cuando y que compras',
    'Tienes bastante GAS, cada vez que hagas una compra intenta que
sea algo que te aporte de verdad. Que no acaben las cosas en un
cajón.',
    'Nivel considerable de GAS, tu verás a donde vas. Piensa bien si
necesitas todo lo que compras']).
gasAnswer(alto, [
    'Pffff vaya manera de despilfarrar el dinero. No seas tonto anda,
padeces de GAS, debes saberlo',
    'Tienes GAS, no hay mas. Intenta no comprar nada en un año, y
vende todo lo que no uses',
    'Anda cambia de hobby. Este te produce demasiado GAS. Acabaras
arruinado.']).
```

.

Como puede verse se hace una unificación en el argumento Grade, haciendo que se selección la regla **getAnswer** correspondiente al nivel de GAS.

4 - COMENTARIO DEL CÓDIGO PROLOG

```
%
-------------------------------------------------------------------------
--------
%
--------------------------------CONSTANTES-------------------------------
--------
%
-------------------------------------------------------------------------
--------
% Niveles de gas
ninguno.
bajo.
medio.
alto.

%
-------------------------------------------------------------------------
--------
%
-------------------------------HECHOS------------------------------------
--------
%
-------------------------------------------------------------------------
--------

% Hechos sin variable
% Este predicado relaciona los niveles de GAS con las respuestas a
dar
gasAnswer(ninguno, [
    'No tienes GAS. Eres un tio con cabeza. Seguro que sabes cuando
darte un capricho y cuando no.',
    'No tienes GAS. Prueba a comprarte algo hombre. Tu no tienes
peligro de caer en el GAS.',
    'No tienes GAS. Enhorabuena. Si quieres comprar algo puedes
hacerlo']).
```

```
gasAnswer(bajo, [
    'Tienes un poco de GAS. No es preocupante, mucha gente le
entusiasma su hobby, mientras no vaya a mas no hay problema.',
    'Apenas tienes GAS, no hay problema mientras no vayas a mas.',
    'No te preocupes no tienes casi GAS, sigue pensando que compras
te aportan algo y cuales no.']).
gasAnswer(medio, [
    'Tu nivel de GAS es preocupante. De aquí a despilfarrar el dinero
hay un paso. Piensa bien cuando y que compras',
    'Tienes bastante GAS, cada vez que hagas una compra intenta que
sea algo que te aporte de verdad. Que no acaben las cosas en un
cajón.',
    'Nivel considerable de GAS, tu verás a donde vas. Piensa bien si
necesitas todo lo que compras']).
gasAnswer(alto, [
    'Pffff vaya manera de despilfarrar el dinero. No seas tonto anda,
padeces de GAS, debes saberlo',
    'Tienes GAS, no hay mas. Intenta no comprar nada en un año, y
vende todo lo que no uses',
    'Anda cambia de hobby. Este te produce demasiado GAS. Acabaras
arruinado.']).


% Hechos con variables dinamicas
:-dynamic pointList/1.
pointList([]).   % Voy almacenando los resulatados en puntos de cada
respuesta

% Representa un punto de la lista de puntos
point(_).

% Lista que almacena las puntuaciones de cada opción de respuesta
questionPoints([10, 50, 100, 500]).

%
--------------------------------------------------------------------
--------
% ----------------------------PREDICADOS
```

```
AUXILIARES---------------------------
%
------------------------------------------------------------------------
--------
% Limpia la pantalla
clearScreen:- write('\033[2J').

% Función que suma los elementos de una lista
sum([],0).
sum([X|C],T):-sum(C,Aux),T is Aux+X.

% Función que devuelñve el elemento N de una lista
getElementList([X|_],1,X).
getElementList([_|Xs],N,Y):- N2 is N-1, getElementList(Xs,N2,Y).

% Función que añade una puntuación a la lista.
insert(X,L,[X|L]).

% Función que borra la lista de puntos de las respuestas
clearPointList:-pointList(PL), retract(pointList(PL)),
asserta(pointList([])).

%
------------------------------------------------------------------------
--------
% ----------------------------PREDICADOS
PRINCIPALES--------------------------
%
------------------------------------------------------------------------
--------

/* Almacena en una lista cada opción respondida en preguntas, si el
número es mayor que 4 se muestra mensaje de error y se repite la
pregunta */
processOption(R,Current):-
    R > 4, nl,nl,nl,tab(5),write('A ver si aparte de GAS tienes algo
mas... Volvemos a empezar. Introduce un numero de 1 al 4
anda'),nl,nl,nl, question(Current);
    R =< 4, pointList(P), point(E), questionPoints(Points),
```

```
        getElementList(Points, R, E), insert(E, P, I),
asserta(pointList(I)), clearScreen.


% Se inicializan las preguntas
question(1):-nl,tab(5),write('En relacion a tus ingresos cuanto
dinero crees que gastas al mes de media?'),nl,
    tab(10),write('1) Menos del 5% del salario'),nl,
    tab(10),write('2) Entre el 5% y el 10%'),nl,
    tab(10),write('3) Entre el 10% y el 15%'),nl,
    tab(10),write('4) Mas del 15%.'),nl,
    nl,nl,tab(5),write('Elija una de las opciones(Recuerde terminar
en punto): '),read(Y),
    processOption(Y, 1).

question(2):-nl,tab(5),write('Cada vez que sale algo nuevo, si es
parecido a lo que ya tienes pero un poco mejor...'),nl,
    tab(10),write('1) Lo miro pero no lo compro. Si no es algo
significativa,emte mejor o que necesite no gasto el dinero'),nl,
    tab(10),write('2) Intento vender lo que ya tengo y si no es mucho
mas lo compro.'),nl,
    tab(10),write('3) Lo compro inmediatamente. Ya intentare vender
lo anterior que tengo.'),nl,
    tab(10),write('4) Lo compro sin pensarmelo dos veces.'),nl,
    nl,nl,tab(5),write('Elija una de las opciones(Recuerde terminar
en punto): '),read(Y),
    processOption(Y, 2).

question(3):-nl,tab(5),write('Cada cuanto frecuentas precia
especializada sobre tu equipo?'),nl,
    tab(10),write('1) Cada mes mas o menos'),nl,
    tab(10),write('2) Todas las semanas.'),nl,
    tab(10),write('3) Todos los dias varias.'),nl,
    tab(10),write('4) Todos los dias varias veces, estoy al tanto de
todo lo que sale'),nl,
    nl,nl,tab(5),write('Elija una de las opciones(Recuerde terminar
en punto): '),read(Y),
    processOption(Y, 3).
```

```
question(4):-nl,tab(5),write('Te arrepientes de cosas que compraste y
ya no usas?'),nl,
    tab(10),write('1) No, no tengo nada que no use'),nl,
    tab(10),write('2) No, suelo vender lo que ya no uso.'),nl,
    tab(10),write('3) Si, hay cosas en las que gaste mucho dinero y
no las aprovecho.'),nl,
    tab(10),write('4) ¿Como quieres que use todo lo que compro?
estamos locos o que.'),nl,
    nl,nl,tab(5),write('Elija una de las opciones(Recuerde terminar
en punto): '),read(Y),
    processOption(Y, 4).

question(5):-nl,tab(5),write('Alguna vez has pasado problemas
economicos por culpa de adquirir equipo para tu hobby?'),nl,
    tab(10),write('1) No, no gasto mas de lo que puedo y siempre
ahorro'),nl,
    tab(10),write('2) No, algun mes me tuve que apretar el cinturon
para darme el capricho, pero nunca me a faltado dinero.'),nl,
    tab(10),write('3) Tengo que pedir prestado a veces.'),nl,
    tab(10),write('4) Estoy endeudado hasta las cejas, pero tengo
muchas cosas, puedo venderlas si no quueda mas remedio.'),nl,
    nl,nl,tab(5),write('Elija una de las opciones(Recuerde terminar
en punto): '),read(Y),
    processOption(Y, 5).

% Este predicado devuelve uno de los grados de GAS existentes en
función de la puntuación obtenida
gasGrade(Num, Grade):-
    Num<250,Grade=ninguno;
    Num>=250,Num<500,Grade=bajo;
    Num>=500,Num<1500,Grade=medio;
    Num>=1500,Grade=alto.


randomAnswer(List, Answer):-length(List, Length), Idx is
random(Length), nth0(Idx, List ,Answer).

% Esta función , para cada tipo de grado elegirá uno al azar de la
lista
```

```
persuadeByGrade(Grade):- gasAnswer(Grade, List), randomAnswer(List,
Answer), nl, write(Answer).

% Le las posibles opciones de repetir o no el test
option(s):- clearPointList, menu.
option(n):- halt.
option(_):- nl, write('Por favor, escriba s o n: '), restart.

% PAra volver al menu inicial una vez finalizado el test o salir
restart:- nl,write('Quiere volver al menu principal s/n?:
'),read(R),option(R).

% Obtiene el consejo a dar al usuario en función del nivel de gas
obtenido como resultado del test así como el número de puntos
getResult:- pointList(Points), sum(Points, Result),
           clearScreen, write('Has obtenido '),write(Result), write('
puntos '),
           gasGrade(Result, Grade), persuadeByGrade(Grade), restart.

startTest:- question(1), question(2), question(3), question(4),
question(5),
           getResult.
% Acciones según respuestas del menú
action(1):- clearScreen, startTest.
action(2):- clearScreen.
action(3):- halt.

% Menu de incio del programa, con las opciones elegidas */
menu:-clearScreen,
   tab(10),write('=============================='),nl,
   clearPointList, tab(10),write('CONSEJOS PARA EVITAR EL GAS'),nl,
    tab(10),write('=============================='),nl,
    tab(15),write('1) Comenzar el test.'),nl,
    tab(15),write('2) Borrar pantalla.'),nl,
    tab(15),write('3) Salir.'),nl,nl,
    tab(10),write('Elija una de las opciones: '),
    read(X), action(X).

% Inicio de el menú
```

```
init:- menu.
```

5 - DESCRIPCIÓN DE LOS CASOS DE PRUEBA

A continuación muestro ejemplos de los requisitos contenidos en la PEC.
Para el inicio del programa es necesario escribir init.

2 o más constantes

```
% CONSTANTES
% Niveles de gas
ninguno.
bajo.
medio.
alto.
```

Ejecución de casos de prueba

```
?- ninguno.
true.

?- bajo.
true.
```

2 o más hechos con variables y 2 o más sin variables

```
% Hechos sin variable
gasNone(['GASNINGUNO1', 'GASNINGUNO2']).
gasLow(['GASBAJO1', 'GASBAJO2', 'GASBAJO3']).
gasMid(['GASMEDIO1', 'GASMEDIO2', 'GASMEDIO3']).
gasHight(['GASALTO1', 'GASALTO2', 'GASALTO3']).

% Voy almacenando los resultados en puntos de cada respuesta
pointList([]).
point(_).
```

Casos de prueba

```
?- gas(X).
```

```
X = bajo .

?- pointList(L).
L = [100, 100]
```

5 o más predicados

```
% Limpia la pantalla
clearScreen:- write('\033[2J').ç

% Función que añade una puntuación a la lista.
insert(X,L,[X|L]).

/* Almacena en una lista cada opción respondida en preguntas, si el número
es mayor que 4 se muestra mensaje de error y se repite la pregunta */
processOption(R,Current):-
    R > 4, nl,tab(5),write('A ver si aparte de GAS tienes algo mas...
Volvemos a empezar. Introduce un numero de 1 al 4 anda'), question(Current);
    R =< 4, pointList(P), point(E), questionPoints(Points),
         getElementList(Points, R, E), insert(E, P, I),
asserta(pointList(I)), clearScreen.

randomAnswer(List, Answer):-length(List, Length), Idx is random(Length),
nth0(Idx, List ,Answer).

/* Menu de incio del programa, con las opciones elegidas */
menu:-clearScreen,
   tab(10),write('============================='),nl,
   clearPointList, nl,tab(10),write('CONSEJOS PARA EVITAR EL GAS'),nl,
    tab(10),write('============================='),nl,
    tab(15),write('1) Comenzar el test.'),nl,
    tab(15),write('2) Borrar pantalla.'),nl,
    tab(15),write('3) Salir.'),nl,nl,
    tab(10),write('Elija una de las opciones: '),
    read(X), action(X).
```

Aquí solo haré uso de los casos de prueba para la función de insert, y de menú:

```
?- insert(120, [1,2], ListaResultado).
ListaResultado = [120, 1, 2].

?- menu.




          ==============================
          CONSEJOS PARA EVITAR EL GAS
          ==============================
               1) Comenzar el test.
               2) Borrar pantalla.
               3) Salir.

          Elija una de las opciones:
```

Al menos uno de los predicados habrá de definirse mediante 2 o más reglas

```
% Función que devuelñve el elemento N de una lista
getElementList([X|_],1,X).
getElementList([_|Xs],N,Y):- N2 is N-1, getElementList(Xs,N2,Y).
```

```
?- getElementList([a,b,c],3,Elemento).
Elemento = c
```

Al menos uno de los predicados deberá tener 2 o más argumentos,

Aquí un ejemplo de predicado con 2 argumentos

```
randomAnswer(List, Answer):-length(List, Length), Idx is
random(Length), nth0(Idx, List ,Answer).
```

```
?- randomAnswer(['opcion1', 'opcion2', 'opcion3'], Respuesta).
Respuesta = opcion3.
```

2 o más reglas deberán constar de 2 o más antecedentes

El predicado getElementList tiene 2 antecedentes en su segunda regla y el de sum tienen 2

```
getElementList([X|_],1,X).
getElementList([_|Xs],N,Y):- N2 is N-1, getElementList(Xs,N2,Y).


sum([],0).
sum([X|C],T):-sum(C,Aux),T is Aux+X.
```

```
?- getElementList([a,b,c],3,Elemento).
Elemento = c
```

La satisfacción de 2 o más objetivos habrá de requerir el encadenamiento de 2 o más reglas.

```
randomAnswer(List, Answer):-length(List, Length), Idx is
random(Length), nth0(Idx, List ,Answer).
```

En el predicado **randomAnswer**, primero obtenemos la longitud de la lista y la guardamos en Lenght, este argumento se encadena con el siguiente antecedente en el predicado random con el fin de que el término Idx almacene un número aleatorio en el rango 0-Length. Finalmente Idx se una en el predicado **nth0** para que nos devuelva el elemento que ocupa la posición Idx en la lista. El argumento List se va encadenando del mismo modo

```
?- randomAnswer(['opcion1', 'opcion2', 'opcion3'], Respuesta).
Respuesta = opcion3.

?- sum([1,2,3], Resultado).
Resultado = 6.
```

Al menos un ejemplo de uso de uno de los operadores aritméticos o relacionales predefinidos.

```
% Esta función devuelve uno de los grados de GAS existentes en función de la puntuación obtenida
gasGrade(Num, Grade):-
    Num=<50,Num>=250,Grade=ninguno;
    Num>=250,Num<500,Grade=bajo;
    Num>=500,Num<1500,Grade=medio;
    Num>=1500,Grade=alto.
```

```
?- gasGrade(1000, Grado).
Grado = alto.
```

Al menos un ejemplo de recursividad

```
sum([],0).
sum([X|C],T):-sum(C,Aux),T is Aux+X.
```

```
?- sum([1,2,3], Resultado).
Resultado = 6.
```

Al menos un ejemplo de uso de los predicados de inserción y borrado de hechos de la Base de Hechos.

```
% Función que añade una puntuación a la lista.
insert(X,L,[X|L]).

/* Función que almacena en una lista cada opción respondida en preguntas */
processOption(R):- pointList(P), point(E), questionPoints(Points),
   getElementList(Points, R, E), insert(E, P, I), asserta(pointList(I)).
```

```
% Función que borra la lista de puntos de las respuestas
clearPointList:-pointList(PL), retract(pointList(PL)),
asserta(pointList([])).
```

A continuación, insertaré en la lista de puntos el valor 33, luego añadiré 55, y despues llamaré al predicado que borra la lista, y volveré a consultar la lista para asegurarme de que es vacía

```
?- pointList(PL), insert(55, PL, PLRES), asserta(pointList(PLRES)).
PL = [],
PLRES = [55].

?- pointList(PL), insert(33, PL, PLRES), asserta(pointList(PLRES)).
PL = [55],
PLRES = [33, 55] .

?- clearPointList.
true .

?- pointList(PL).
PL = []
```

6 - DIFICULTADES ENCONTRADAS.

Las mayores dificultades las he encontrado en la elección del tema, de entre tantos disponibles y tan genéricos, y una vez elegido uno de los temas (el consumismo en mi caso), concretar con el desarrollo de una práctica basado en aquello que los guitarristas (y otros) llamamos GAS.

Otra gran dificultad fué el aprendizaje básico del lenguaje Prolog. Es muy diferente de la programación por procedimientos o la programación orientada a objetos, que es a la que estoy acostumbrado yo.

Me costó bastante enterarme de cómo se manejan las listas en Prolog, y entender cómo funcionaba el insertado y borrado en ellas.