MEMORIA PECL1 CRA

¿Quién es quién?

23 DE MARZO DE 2021

ROBERT PETRISOR X9441429K
DAVID MÁRQUEZ MÍNGUEZ 47319570Z

EXPLICACIÓN DE CODIGO

Para empezar, vamos a hacer una pequeña introducción al modo de juego ¿Quién es quién?. ¿Quién es quién? es un juego de mesa consistente en dos tableros idénticos en los que aparece un cierto número de personajes identificados por su nombre, cada uno con sus características físicas particulares. Cada jugador recibe un personaje distinto y el contrario tiene que adivinarlo mediante sencillas preguntas triviales. Las preguntas se hacen por turnos y gana el que primero adivine el personaje del contrario.

En cuanto al número de personajes identificados por su nombre, corresponde a los definidos tal como viene en el enunciado de la práctica junto con la adicción de una pequeña porción hasta llegar al valor de 30 personajes diferentes. Mediante el cual podemos observar en el siguiente listado: Albert, Paul, Tom, Richard, Louis, Michael, Charles, Sam, Steve, Will, Anthony, Billy, Henry, Tiffany, Natalie, Roxanne, Sarah, Sabrina, Cindy, Emma, David, John, Mike, Damon, Christian, Cloe, Sophia, Emily, Rose y Lia.

Las características definidas de cada personaje son:

```
%personaje(nombre, [genero, color pelo, color ropa, estado animico, gafas?, color ojos, edad, sombrero?, barba?]).
```

Podemos ver la definición concreta en el siguiente código:

```
personaje('Albert', ['masculino', 'pelo_negro', 'ropa_roja', 'feliz', 'con_gafas', 'ojos_azules', 'joven', 'sin_sombrero', 'con_barba']).
personaje('Paul', ['masculino', 'pelo_rubio', 'ropa_roja', triste, 'sin_gafas', 'ojos_marrones', 'anciano', 'sin_sombrero', 'sin_barba']).
personaje('Tom', ['masculino', 'pelo_negro', 'ropa_verde', 'feliz', 'sin_gafas', 'ojos_marrones', 'joven', 'sin_sombrero', 'con_barba']).
personaje('Richard', ['masculino', 'pelo_negro', 'ropa_verde', 'triste', 'sin_gafas', 'ojos_marrones', 'joven', 'con_sombrero', 'con_barba']).
personaje('Louis', ['masculino', 'pelo_negro', 'ropa_roja', 'triste', 'sin_gafas', 'ojos_azules', 'joven', 'sin_sombrero', 'con_barba']).
personaje('Michael', ['masculino', 'pelo_rubio', 'ropa_verde', 'feliz', 'con_gafas', 'ojos_marrones', 'anciano', 'sin_sombrero', 'sin_barba']).
personaje('Charles', ['masculino', 'pelo_negro', 'ropa_verde', 'feliz', 'con_gafas', 'ojos_marrones', 'joven', 'sin_sombrero', 'sin_barba']).
personaje('Sam', ['masculino', 'pelo_rubio', 'ropa_roja', 'triste', 'sin_gafas', 'ojos_azules', 'anciano', 'con_sombrero', 'con_barba']).
personaje('Steve', ['masculino', 'pelo_negro', 'ropa_roja', 'feliz', 'sin_gafas', 'ojos_marrones', 'anciano', 'con_sombrero', 'con_barba']).
personaje('Will', ['masculino', 'pelo_rubio', 'ropa_verde', 'feliz', 'sin_gafas', 'ojos_azules', 'joven', 'sin_sombrero', 'sin_barba']).
personaje('Anthony', ['masculino', 'pelo_rubio', 'ropa_roja', 'feliz', 'con_gafas', 'ojos_marrones', 'joven', 'sin_sombrero', 'sin_barba']).
personaje('Billy', ['masculino', 'pelo_rubio', 'ropa_verde', 'triste', 'sin_gafas', 'ojos_azules', 'anciano', 'sin_sombrero', 'con_barba']).
personaje('Henry', ['masculino', 'pelo_negro', 'ropa_roja', 'triste', 'sin_gafas', 'ojos_marrones', 'joven', 'con_sombrero', 'sin_barba']).
personaje('Tiffany', ['femenino', 'pelo_negro', 'ropa_verde', 'feliz', 'sin_gafas', 'ojos_marrones', 'joven', 'sin_sombrero', 'sin_barba']).
personaje('Natalie', ['femenino', 'pelo_rubio', 'ropa_roja', 'feliz', 'con_gafas', 'ojos_azules', 'joven', 'sin_sombrero', 'sin_barba']).
personaje('Roxanne', ['femenino', 'pelo_rubio', 'ropa_verde', 'feliz', 'sin_gafas', 'ojos_azules', 'anciano', 'sin_sombrero', 'sin_barba']).
personaje('Sarah', ['femenino', 'pelo_negro', 'ropa_roja', 'triste', 'sin_gafas', 'ojos_marrones', 'anciano', 'sin_sombrero', 'sin_barba']).
personaje('Sabrina', ['femenino', 'pelo_negro', 'ropa_verde', 'feliz', 'con_gafas', 'ojos_azules', 'anciano', 'sin_sombrero', 'sin_barba']).
personaje('Cindy', ['femenino', 'pelo_negro', 'ropa_roja', 'feliz', 'sin_gafas', 'ojos_marrones', 'joven', 'con_sombrero', 'sin_barba']).
personaje('Emma', ['femenino', 'pelo_rubio', 'ropa_verde', 'feliz', 'sin_gafas', 'ojos_marrones', 'joven', 'con_sombrero', 'sin_barba']).
personaje('David', ['masculino', 'pelo_rubio', 'ropa_roja', 'feliz', 'con_gafas', 'ojos_azules', 'joven', 'sin_sombrero', 'sin_barba']).
personaje('John', ['masculino', 'pelo_rubio', 'ropa_verde', 'feliz', 'con_gafas', 'ojos_azules', 'joven', 'con_sombrero', 'sin_barba']).
personaje('Mike', ['masculino', 'pelo_rubio', 'ropa_verde', 'triste', 'con_gafas', 'ojos_azules', 'joven', 'con_sombrero', 'sin_barba']).
personaje('Damon', ['masculino', 'pelo_rubio', 'ropa_roja', 'feliz', 'con_gafas', 'ojos_azules', 'anciano', 'sin_sombrero', 'sin_barba']).
personaje('Christian', ['masculino', 'pelo_negro', 'ropa_verde', 'feliz', 'con_gafas', 'ojos_azules', 'joven', 'sin_sombrero', 'con_barba']).
personaje('Cloe', ['femenino', 'pelo_negro', 'ropa_verde', 'triste', 'sin_gafas', 'ojos_azules', 'anciano', 'con_sombrero', 'sin_barba']).
personaje('Sophia', ['femenino', 'pelo_rubio', 'ropa_roja', 'triste', 'con_gafas', 'ojos_azules', 'joven', 'con_sombrero', 'sin_barba']).
personaje('Emily', ['femenino', 'pelo_rubio', 'ropa_roja', 'triste', 'con_gafas', 'ojos_azules', 'joven', 'con_sombrero', 'sin_barba']).
personaje('Rose', ['femenino', 'pelo_negro', 'ropa_verde', 'triste', 'con_gafas', 'ojos_marrones', 'joven', 'con_sombrero', 'sin_barba']).
personaje('Lia', ['femenino', 'pelo_negro', 'ropa_verde', 'triste', 'con_gafas', 'ojos_marrones', 'anciano', 'sin_sombrero', 'sin_barba']).
```

En **listado_personajes.pl**: viene definido dicha base de datos con los personajes junto con listarPersonajes, que viene siendo la lista formada por el nombre de cada uno de los personajes de dicha base de datos.

```
listarPersonajes(ListaPersonajes):- listarAux([], ListaPersonajes),!.
listarAux(Y,Z):- personaje(Personaje,_), not(member(Personaje,Y)), listarAux([Personaje|Y],Z).
listarAux(X,X).
```

Una vez definido los personajes, empezamos a plantear las siguientes preguntas acerca de las características personales de cada uno de los personajes descritos anteriormente. Que, por carácter general, las preguntas van por las siguientes direcciones:

1. ¿Chico? o ¿Chica? -> Sobre el género

```
es_chico(Lista,ListaFinalOut,L1,L2,Personaje):- procesar_opcion_es_chico(Personaje,Lista,[],ListaFinalOut,L1,L2).
es_chica(Lista,ListaFinalOut,L1,L2,Personaje):- procesar_opcion_es_chica(Personaje,Lista,[],ListaFinalOut,L1,L2).
```

2. ¿Pelo negro? o ¿Pelo rubio? -> Sobre el color del pelo

```
pelo_negro(Lista,ListaFinalOut,L1,L2,Personaje):- procesar_opcion_pelo_negro(Personaje,Lista,[],ListaFinalOut,L1,L2).
pelo_rubio(Lista,ListaFinalOut,L1,L2,Personaje):- procesar_opcion_pelo_rubio(Personaje,Lista,[],ListaFinalOut,L1,L2).
```

3. ¿Ropa roja? O ¿Ropa verde? -> Sobre el color de la ropa

```
ropa_roja(Lista,ListaFinalOut,L1,L2,Personaje):- procesar_opcion_ropa_roja(Personaje,Lista,[],ListaFinalOut,L1,L2).
ropa_verde(Lista,ListaFinalOut,L1,L2,Personaje):- procesar_opcion_ropa_verde(Personaje,Lista,[],ListaFinalOut,L1,L2).
```

4. ¿Feliz? o ¿Triste? -> Sobre el estado anímico

```
feliz(Lista,ListaFinalOut,L1,L2,Personaje):- procesar_opcion_feliz(Personaje,Lista,[],ListaFinalOut,L1,L2).
triste(Lista,ListaFinalOut,L1,L2,Personaje):- procesar_opcion_feliz(Personaje,Lista,[],ListaFinalOut,L1,L2).
```

5. ¿Gafas? -> Sobre las gafas (sin/con)

```
gafas(Lista,ListaFinalOut,L1,L2,Personaje):- procesar_opcion_gafas(Personaje,Lista,[],ListaFinalOut,L1,L2).
```

6. ¿Ojos azules? o ¿Ojos marrones? -> Sobre el color de los ojos

```
ojos_azules(Lista,ListaFinalOut,L1,L2,Personaje):- procesar_opcion_ojos_azules(Personaje,Lista,[],ListaFinalOut,L1,L2).
ojos_marrones(Lista,ListaFinalOut,L1,L2,Personaje):- procesar_opcion_ojos_marrones(Personaje,Lista,[],ListaFinalOut,L1,L2).
```

7. ¿Joven? O ¿Anciano? -> Sobre la edad

```
joven(Lista,ListaFinalOut,L1,L2,Personaje):- procesar_opcion_joven(Personaje,Lista,[],ListaFinalOut,L1,L2).
anciano(Lista,ListaFinalOut,L1,L2,Personaje):- procesar_opcion_anciano(Personaje,Lista,[],ListaFinalOut,L1,L2).
```

8. ¿Sombrero? -> Sobre el sombrero (sin/con)

```
con_sombrero(Lista,ListaFinalOut,L1,L2,Personaje):- procesar_opcion_con_sombrero(Personaje,Lista,[],ListaFinalOut,L1,L2).
sin_sombrero(Lista,ListaFinalOut,L1,L2,Personaje):- procesar_opcion_sin_sombrero(Personaje,Lista,[],ListaFinalOut,L1,L2).
```

9. ¿Barba? -> Sobre la barba (sin/con)

```
barba(Lista,ListaFinalOut,L1,L2,Personaje):- procesar_opcion_barba(Personaje,Lista,[],ListaFinalOut,L1,L2).
```

Con todas estas preguntas es posible diferenciar a todos los personajes. Gran parte del trabajo lo realizan las preguntas generales y amplias en cuanto a algunos rasgos distintivos de ciertos personajes. Dado que, en un momento en concreto el conjunto de preguntas, consiguen discriminar las opciones restantes.

Toda la información correspondiente al listado de preguntas a realizar viene dada en el fichero: **listado_preguntas.pl**. Cabe destacar que se incluye en una lista llamada listarPreguntas a la lista creada con el listado de las posibles preguntas a establecer.

```
listarPreguntas(ListaPreguntas):- ListaPreguntas = ['es_chico','es_chica','pelo_negro','pelo_rubio','ropa_verde','ropa_roja',
        'feliz','triste','gafas','ojos_azules','ojos_marrones','joven','anciano','con_sombrero','sin_sombrero','barba'].
```

Ahora vamos a explicar el fichero **main.pl**, mediante el cual contiene la lógica de ejecución de la aplicación. En cuanto al funcionamiento general, destacamos los siguientes pasos:

- Lanzar el menú y preguntar si se quiere jugar y dentro de eso, que escoja una posible opción de juego (jugador vs maquina normal / jugador vs maquina avanzado / jugador vs jugador). En caso de que no querer jugar, salir del programa.

```
jugar:- write('________            .__                 ________            .__                    '), nl,
        write('\_____  \  __ __|__| ____   ____   ____  ______\_____  \  __ __|__| ____   ____   '), nl,
        write(' /  / \  \|  |  \  |/ __ \ /    \ _/ __ \ /  ___/ /  / \  \|  |  \  |/ __ \ /    \ '), nl,
        write('/   \_/.  \  |  /  \  ___/|   |  \\  ___/ \___ \ /   \_/.  \  |  /  \  ___/|   |  \'), nl,
        write('\_____\ \_/____/|__|\___  >___|  /  \___  >____  >\_____\ \_/____/|__|\___  >___|  /'), nl,
        write('       \__>              \/     \/       \/     \/        \__>             \/     \/ '), nl,
        writeln(' '), nl,
        write('Escribe 1. para seleccionar el modo jugador vs maquina normal'), nl,
        write('Escribe 2. para seleccionar el modo jugador vs maquina avanzado'), nl,
        write('Escribe 3. para seleccionar el modo jugador vs jugador'), nl,
        write('Escribe 4. para salir'), nl,
        read(X),
        comprobar_respuesta(X).
```

- Una vez escogida la opción de juego, leemos el archivo de personajes y cargamos en la lista de personajes posibles, y se asigna un personaje a su correspondiente jugador/máquina.
- Una vez asignado el personaje, mostramos sus características de dicho personaje tanto para el jugador como la máquina (para que cada jugador/maquina sepa las características del personaje escogido).
- Listamos la lista de los personajes y la lista de las preguntas para cada jugador/máquina correspondiente para preparar el dicho juego.
- Por último, llamamos al siguiente método con una de las siguientes posibilidades del modo de juego escogido:

```
jugador_vs_maquina(PersonajeJugador,PersonajeMaquina,ListaPersonajesJugador,ListaPersonajesMaquina,ListaPreguntasJugador,ListaPreguntasMaquina,jugador).

jugador_vs_maquina_avanzado(PersonajeJugador,PersonajeMaquina,ListaPersonajesJugador,ListaPersonajesMaquina,ListaPreguntasJugador,ListaPreguntasMaquina,jugador).

jugador_vs_jugador(PersonajeJugador1,PersonajeJugador2,ListaPersonajesJugador1,ListaPersonajesJugador2,ListaPreguntasJugador1,ListaPreguntasJugador2,jugador1).
```

En el cual está implementado en los ficheros correspondientes: juego_normal.pl y juego_avanzado.pl. En ellos contiene la lógica de la aplicación del modo de juego de la práctica.

Ejemplo: si escogemos la opción 1, tendríamos el siguiente código necesario:

```
comprobar_respuesta(1):- listarPersonajes(ListaPersonajes),
                         write('La lista de personajes es: '),
                         writeln(ListaPersonajes),
                         writeln(''),
                         asignar_personaje(PersonajeJugador),
                         asignar_personaje(PersonajeMaquina),
                         write('Tu personaje asignado es: '),
                         writeln(PersonajeJugador),
                         personaje(PersonajeJugador,CaracteristicasPersonajeJugador),
                         write('Sus carcateristicas son: '),
                         writeln(CaracteristicasPersonajeJugador),
                         write('El personaje de la maquina es: '),
                         writeln(PersonajeMaquina),
                         listarPreguntas(ListaPreguntasJugador), listarPreguntas(ListaPreguntasMaquina),
                         listarPersonajes(ListaPersonajesJugador), listarPersonajes(ListaPersonajesMaquina),
                         jugador_vs_maquina(PersonajeJugador,PersonajeMaquina,ListaPersonajesJugador,ListaPersonajesMaquina,ListaPreguntasJugador,ListaPreguntasMaquina,jugador).
```

Para la opción 2:

```
comprobar_respuesta(2):- listarPersonajes(ListaPersonajes),
                         write('La lista de personajes es: '),
                         writeln(ListaPersonajes),
                         writeln(''),
                         asignar_personaje(PersonajeJugador),
                         asignar_personaje(PersonajeMaquina),
                         write('Tu personaje asignado es: '),
                         writeln(PersonajeJugador),
                         personaje(PersonajeJugador,CaracteristicasPersonajeJugador),
                         write('Sus carcateristicas son: '),
                         writeln(CaracteristicasPersonajeJugador),
                         write('El personaje de la maquina es: '),
                         writeln(PersonajeMaquina),
                         listarPreguntas(ListaPreguntasJugador),
                         listarPersonajes(ListaPersonajesJugador), listarPersonajes(ListaPersonajesMaquina),
                         seleccionar_mejor_pregunta(ListaPersonajes,ListaPreguntasMaquina),
                         jugador_vs_maquina_avanzado(PersonajeJugador,PersonajeMaquina,ListaPersonajesJugador,ListaPersonajesMaquina,ListaPreguntasJugador,ListaPreguntasMaquina,jugador).
```

Para la opción 3:

```
comprobar_respuesta(3):- listarPersonajes(ListaPersonajes),
                         write('La lista de personajes es: '),
                         writeln(ListaPersonajes),
                         writeln(''),
                         asignar_personaje(PersonajeJugador1),
                         asignar_personaje(PersonajeJugador2),
                         write('El personaje del jugador1 es: '),
                         writeln(PersonajeJugador1),
                         personaje(PersonajeJugador1,CaracteristicasPersonajeJugador1),
                         write('Sus carcateristicas son: '),
                         writeln(CaracteristicasPersonajeJugador1),
                         write('El personaje del jugador2 es: '),
                         writeln(PersonajeJugador2),
                         listarPreguntas(ListaPreguntasJugador1), listarPreguntas(ListaPreguntasJugador2),
                         listarPersonajes(ListaPersonajesJugador1), listarPersonajes(ListaPersonajesJugador2),
                         jugador_vs_jugador(PersonajeJugador1,PersonajeJugador2,ListaPersonajesJugador1,ListaPersonajesJugador2,ListaPreguntasJugador1,ListaPreguntasJugador2,jugador1).
```

Para la opción 4:

```
comprobar_respuesta(4):- write('Hasta luego :(').
```

Cabe destacar que para el modo avanzado de la máquina se llamará a la siguiente función que se explicará más con detalle más en delante de este documento:

```
seleccionar_mejor_pregunta(ListaPersonajes,ListaPreguntasMaquina),
```

Para que, de esta forma, en vez de escoger al azar una de entre las no realizadas hasta el momento de la lista de preguntas a hacer, la máquina aplique un algoritmo que le permita ganar la partida con el número mínimo de preguntas posibles.

Ahora solo nos queda explicar el funcionamiento de los dos ficheros: **juego_normal.pl** y **juego_avanzado.pl**. Comencemos primero por el **juego_nomal.pl**:

Viene divida en tres secciones: una para el procesamiento de los turnos entre jugador y jugador/maquina, otra para las principales funciones necesarias para el programa, y la otra para el procesamiento de las preguntas a realizar.

- En la primera de ellas (**procesamiento de turnos**):

En caso del procesamiento de turnos durante una partida de jugador vs máquina, sería de la siguiente forma:

```
jugador_vs_maquina(PersonajeJugador,PersonajeMaquina,ListaPersonajesJugador,ListaPersonajesMaquina,ListaPreguntasJugador,ListaPreguntasMaquina,Turno):-
                   %Se comprueba que si la partida ha acabado o no
                   length(ListaPersonajesJugador,Tam), Tam==1 -> nl, write('He ganado, se que eres '), write(ListaPersonajesJugador);
                   length(ListaPersonajesMaquina,Tam), Tam==1  -> nl, write('Tu ganas, ya sabias que era '), write(ListaPersonajesMaquina);

                   %Turno del jugador
                   Turno==jugador -> turno_jugador(PersonajeJugador,PersonajeMaquina,ListaPersonajesJugador,ListaPersonajesMaquina,ListaPreguntasJugador,ListaPreguntasMaquina);

                   %Turno de la maquina
                   Turno==maquina -> turno_maquina(PersonajeJugador,PersonajeMaquina,ListaPersonajesJugador,ListaPersonajesMaquina,ListaPreguntasJugador,ListaPreguntasMaquina);

                   writeln('Error en los turnos').
```

Donde básicamente llamamos al método turno_jugador o turno_máquina, para que simulen el intercambio de turnos, hasta que llegamos a la situación final donde unos de los dos (jugador o máquina) queda con una opción de decir el personaje elegido del rival.

En el caso del procesamiento de turnos durante una partida de jugador vs jugador, sería de la siguiente forma:

```
jugador_vs_jugador(PersonajeJugador1,PersonajeJugador2,ListaPersonajesJugador1,ListaPersonajesJugador2,ListaPreguntasJugador1,ListaPreguntasJugador2,Turno):-
        %Se comprueba que si la partida ha acabado o no
        length(ListaPersonajesJugador1,Tam),Tam==1 -> nl, writeln('Gana el jugador2!!');
        length(ListaPersonajesJugador2, Tam),Tam==1  -> nl, writeln('Gana el jugador1!!');

        %Turno de uno de los jugadores
        Turno==jugador1 -> writeln(' '),nl,
                           write('Turno de '), write(Turno), nl,
                           writeln('Elige de entre las siguientes preguntas una que quieras hacerme y escríbela con un punto al final: '),
                           writeln(ListaPreguntasJugador1),
                           read(X), %El jugador hace la pregunta
                           pregunta_disponible(PersonajeJugador1,PersonajeJugador2,ListaPersonajesJugador1,ListaPersonajesJugador2,ListaPreguntasJugador1,ListaPreguntasJugador2,Turno,X,ListaPreguntasJugador1Out),
                           procesar_pregunta(X,ListaPersonajesJugador2,PersonajeJugador2,ListaFinalOut), %La pregunta se procesa y la maquina contesta
                           length(ListaFinalOut,Length),
                           write('Me quedan '), write(Length), write(' fichas aun.'),
                           jugador_vs_jugador(PersonajeJugador1,PersonajeJugador2,ListaPersonajesJugador1,ListaFinalOut,ListaPreguntasJugador1Out,ListaPreguntasJugador2,jugador2);

        %Turno del otro jugador
        Turno==jugador2 -> writeln(' '),nl,
                           write('Turno de '), write(Turno), nl,
                           writeln('Elige de entre las siguientes preguntas una que quieras hacerme y escríbela con un punto al final: '),
                           writeln(ListaPreguntasJugador2),
                           read(X), %El jugador hace la pregunta
                           pregunta_disponible(PersonajeJugador1,PersonajeJugador2,ListaPersonajesJugador1,ListaPersonajesJugador2,ListaPreguntasJugador1,ListaPreguntasJugador2,Turno,X,ListaPreguntasJugador2Out),
                           procesar_pregunta(X,ListaPersonajesJugador1,PersonajeJugador1,ListaFinalOut), %La pregunta se procesa y la maquina contesta
                           length(ListaFinalOut,Length),
                           write('Me quedan '), write(Length), write(' fichas aun.'),
                           jugador_vs_jugador(PersonajeJugador1,PersonajeJugador2,ListaFinalOut,ListaPersonajesJugador2,ListaPreguntasJugador1,ListaPreguntasJugador2Out,jugador1).
```

Similar al caso anterior, lo único que simulan el intercambio de turnos entre dos jugadores.

Para entrar más en detalle, dentro de cada método turno_jugador o turno_máquina, lo vemos a continuación:

```
turno_jugador(PersonajeJugador,PersonajeMaquina,ListaPersonajesJugador,ListaPersonajesMaquina,ListaPreguntasJugador,ListaPreguntasMaquina):-
        writeln(' '),nl,
        writeln('Es tu turno.'),
        writeln('Elige de entre las siguientes preguntas una que quieras hacerme y escríbela con un punto al final: '),
        writeln(ListaPreguntasJugador),
        read(X), %El jugador hace la pregunta
        %Se determina si la pregunta esta disponible
        pregunta_disponible(PersonajeJugador,PersonajeMaquina,ListaPersonajesJugador,ListaPersonajesMaquina,ListaPreguntasJugador,ListaPreguntasMaquina,X,ListaPreguntasJugadorOut),
        %Se procesa la pregunta
        procesar_pregunta(X,ListaPersonajesMaquina,PersonajeMaquina,ListaFinalOut), %La pregunta se procesa y la maquina contesta
        length(ListaFinalOut,Length),
        write('Me quedan '), write(Length), write(' fichas aun.'),
        jugador_vs_maquina(PersonajeJugador,PersonajeMaquina,ListaPersonajesJugador,ListaFinalOut,ListaPreguntasJugadorOut,ListaPreguntasMaquina,maquina).
```

```
turno_maquina(PersonajeJugador,PersonajeMaquina,ListaPersonajesJugador,ListaPersonajesMaquina,ListaPreguntasJugador,ListaPreguntasMaquina):-
        writeln(' '),nl,
        write('Ahora te hago yo una pregunta: '),
        seleccionar_pregunta_aleatoria(X,ListaPreguntasMaquina,ListaPreguntasMaquinaOut),
        writeln(' '),
        procesar_pregunta(X,ListaPersonajesJugador,PersonajeJugador,ListaFinalOut),
        length(ListaFinalOut,Length),
        write('Aun dudo entre '), write(Length), write(' posibilidades.'),
        jugador_vs_maquina(PersonajeJugador,PersonajeMaquina,ListaFinalOut,ListaPersonajesMaquina,ListaPreguntasJugador,ListaPreguntasMaquinaOut,jugador).
```

Como vemos a continuación, en cada turno tanto jugador como máquina, se selecciona una pregunta solicitada (en caso de la máquina modo normal, escoge una pregunta aleatoria de la lista de preguntas), vemos si está disponible, en caso afirmativo, procesamos la pregunta, y analizamos las posibles soluciones de quien personaje puede tratarse. Después cedemos el turno al otro jugador/máquina.

- En la segunda de ellas (**funciones** básicas necesarias):

Tales como:

- Asignar un personaje de forma aleatoria:

```
asignar_personaje(Personaje):- listarPersonajes(ListaPersonajes),
                               random_select(Personaje,ListaPersonajes,_)
```

- Eliminación de un elemento de la lista:

```
del(X,[X|Tail],Tail).
del(X,[Head|Tail],[Head|NewTail]):- del(X,Tail,NewTail).
```

- Filtración de una lista dada un filtro determinado debido a la característica de un personaje:

```
filtrar_lista([],_,ListaFinal,ListaFinal).
filtrar_lista(Lista,Filtro,ListaFinal,ListaFinalOut):- personaje(X,Z), member(Filtro,Z), member(X, Lista), not(member(X, ListaFinal)),
                                                       append(ListaFinal, [X] , NuevaListaFinal),
                                                       subtract(Lista,[X],NuevaLista),
                                                       filtrar_lista(NuevaLista,Filtro,NuevaListaFinal, ListaFinalOut);
                                                       subtract(Lista,[_],NuevaLista),
                                                       filtrar_lista(NuevaLista,Filtro,ListaFinal, ListaFinalOut).
```

- Procesamiento de la pregunta realizada (en caso contrario, error):

```
procesar_pregunta(X,ListaPersonajes,Personaje,ListaFinalOut):-
                  X==es_chico -> es_chico(ListaPersonajes,ListaFinalOut,[],_,Personaje);
                  X==es_chica -> es_chica(ListaPersonajes,ListaFinalOut,[],_,Personaje);
                  X==pelo_negro -> pelo_negro(ListaPersonajes,ListaFinalOut,[],_,Personaje);
                  X==pelo_rubio -> pelo_rubio(ListaPersonajes,ListaFinalOut,[],_,Personaje);
                  X==ropa_verde -> ropa_verde(ListaPersonajes,ListaFinalOut,[],_,Personaje);
                  X==ropa_roja -> ropa_roja(ListaPersonajes,ListaFinalOut,[],_,Personaje);
                  X==feliz -> feliz(ListaPersonajes,ListaFinalOut,[],_,Personaje);
                  X==triste -> triste(ListaPersonajes,ListaFinalOut,[],_,Personaje);
                  X==gafas -> gafas(ListaPersonajes,ListaFinalOut,[],_,Personaje);
                  X==ojos_azules -> ojos_azules(ListaPersonajes,ListaFinalOut,[],_,Personaje);
                  X==ojos_marrones -> ojos_marrones(ListaPersonajes,ListaFinalOut,[],_,Personaje);
                  X==joven -> joven(ListaPersonajes,ListaFinalOut,[],_,Personaje);
                  X==anciano -> anciano(ListaPersonajes,ListaFinalOut,[],_,Personaje);
                  X==con_sombrero -> con_sombrero(ListaPersonajes,ListaFinalOut,[],_,Personaje);
                  X==sin_sombrero -> sin_sombrero(ListaPersonajes,ListaFinalOut,[],_,Personaje);
                  X==barba -> barba(ListaPersonajes,ListaFinalOut,[],_,Personaje);

                  writeln('Error al escribir la pregunta.'),
                  fail.
```

- Determinar si una pregunta está disponible o no (para evitar duplicidad de preguntas):

```
pregunta_disponible(PersonajeJugador,PersonajeMaquina,ListaPersonajesJugador,ListaPersonajesMaquina,ListaPreguntasJugador,ListaPreguntasMaquina,Pregunta,ListaPreguntasJugadorOut):-
                   %Si la pregunta es cuestion se encuentra dentro de la lista de pregunta, se elimina
                   member(Pregunta,ListaPreguntasJugador) -> del(Pregunta,ListaPreguntasJugador,ListaPreguntasJugadorOut);

                   writeln('Debes seleccionar una pregunta valida!'),
                   turno_jugador(PersonajeJugador,PersonajeMaquina,ListaPersonajesJugador,ListaPersonajesMaquina,ListaPreguntasJugador,ListaPreguntasMaquina).

pregunta_disponible(PersonajeJugador1,PersonajeJugador2,ListaPersonajesJugador1,ListaPersonajesJugador2,ListaPreguntasJugador1,ListaPreguntasJugador2,Turno,Pregunta,ListaPreguntasJugadorOut):-
                   %Dependiendo dle turno del jugador, se determina si la pregunta esta disponible o no
                   Turno==jugador1, member(Pregunta,ListaPreguntasJugador1) -> del(Pregunta,ListaPreguntasJugador1,ListaPreguntasJugadorOut);
                   Turno==jugador2, member(Pregunta,ListaPreguntasJugador2) -> del(Pregunta,ListaPreguntasJugador2,ListaPreguntasJugadorOut);

                   writeln('Debes seleccionar una pregunta valida!'),
                   jugador_vs_jugador(PersonajeJugador1,PersonajeJugador2,ListaPersonajesJugador1,ListaPersonajesJugador2,ListaPreguntasJugador1,ListaPreguntasJugador2,Turno).
```

- Procesamiento de la respuesta según lo recibido:

```
procesar_respuesta(ListaFinalOut,Respuesta):-
                                              %Si la respuesta a la pregunta es afirmativa
                                              Respuesta==1 -> writeln('La respuesta a la pregunta es afirmativa'),
                                              writeln('Puede ser uno de los personajes de esta lista: '),
                                              writeln(ListaFinalOut);

                                              %Si la respuesta  la pregunta es negativa
                                              Respuesta==2 -> writeln('La respuesta a la pregunta es negativa'),
                                              writeln('Puede ser uno de los personajes de esta lista: '),
                                              writeln(ListaFinalOut).
```

- Seleccionar de forma aleatoria del grupo de preguntas disponibles:

```
seleccionar_pregunta_aleatoria(X,ListaPreguntas,ListaPreguntasOut):- random_member(X,ListaPreguntas),
                                                                     write(X),
                                                                     del(X,ListaPreguntas,ListaPreguntasOut).
```

- En la tercera de ellas (**procesamiento de preguntas**):

Debido a gran cantidad de preguntas, vamos a ver tres ejemplos en concreto (son similares para el resto de las preguntas procesadas):

```
procesar_opcion_es_chico(Personaje,Lista,ListaFinal,ListaFinalOut,L1,L2):- personaje(Personaje,Z), member('masculino',Z) -> filtrar_lista(Lista,'masculino',ListaFinal,ListaFinalOut),
                                                                           append(L1,['masculino'],L2),
                                                                           procesar_respuesta(ListaFinalOut,1);

                                                                           filtrar_lista(Lista,'femenino',ListaFinal,ListaFinalOut),
                                                                           append(L1,['femenino'],L2),
                                                                           procesar_respuesta(ListaFinalOut,2).

procesar_opcion_pelo_negro(Personaje,Lista,ListaFinal,ListaFinalOut,L1,L2):- personaje(Personaje,Z), member('pelo_negro',Z) -> filtrar_lista(Lista,'pelo_negro',ListaFinal,ListaFinalOut),
                                                                             append(L1,['pelo_negro'],L2),
                                                                             procesar_respuesta(ListaFinalOut,1);

                                                                             filtrar_lista(Lista,'pelo_rubio',ListaFinal,ListaFinalOut),
                                                                             append(L1,['pelo_rubio'],L2),
                                                                             procesar_respuesta(ListaFinalOut,2).

procesar_opcion_ropa_roja(Personaje,Lista,ListaFinal,ListaFinalOut,L1,L2):- personaje(Personaje,Z), member('ropa_roja',Z) -> filtrar_lista(Lista,'ropa_roja',ListaFinal,ListaFinalOut),
                                                                            append(L1,['ropa_roja'],L2),
                                                                            procesar_respuesta(ListaFinalOut,1);

                                                                            filtrar_lista(Lista,'ropa_verde',ListaFinal,ListaFinalOut),
                                                                            append(L1,['ropa_verde'],L2),
                                                                            procesar_respuesta(ListaFinalOut,2).
```

Estas son funciones utilizadas para filtrar las respuestas a las preguntas, de manera que de forma general:

- Las respuestas complementarias suelen filtrar información de la lista de personajes o nos permiten excluir o tomar el contrario, pasándole a la siguiente pregunta el resultado filtrado.

- Ambas respuestas suelen añadir características a la lista de características, de manera que hará que se produzca de manera rápida la adivinación del personaje tanto por parte del jugador como de la máquina.

- Todo esto funciona gracias a: que todas las respuestas son complementarias, es decir: uno o el complementario. Por lo que no hay ambigüedades, por eso siempre llegamos a solución final. Además, tenemos controlado el flujo de preguntas, lo cual nos da precisión y rapidez a la hora de determinar el personaje a lo largo del tiempo.

En cuanto al fichero juego_avanzado.pl: es más corto de codificación ya que solo consta de dos secciones: una dedicada para el procesamiento de turnos y la otra para las funciones básicas.

En cuanto al procesamiento de turnos, es exactamente igual que la codificación del juego normal, lo único que cambia es el tipo de modo al avanzado, donde en vez de seleccionarse las preguntas de forma aleatoria se sigue una lógica que se explica a continuación.

La lógica es la siguiente: a cada una de las preguntas se le asigna una puntuación dependiendo del número de personajes a los que interfiera. Por ejemplo, si para la pregunta es_chico?, se filtran 9 personajes, se asigna a dicha pregunta un valor de 9. Una vez se almacenan todas las puntuaciones de las preguntas, se seleccionan aquellas con menor puntuación. De esta forma, cuando se selecciona una pregunta con la menor puntación posible, la respuesta será la menor cantidad de personajes posibles. Como es lógico el número de preguntas a realizar hasta encontrar al jugador final será el mínimo.

En cuanto a las funciones destacamos las siguientes:

- Seleccionar_mejor_pregunta:

```
seleccionar_mejor_pregunta(ListaPersonajes,ListaPreguntasDisponiblesOut):-

        filtrar_lista(ListaPersonajes,'masculino',[],ListaOutMaso), length(ListaOutMaso,TamListaMaso),
        filtrar_lista(ListaPersonajes,'femenino',[],ListaOutFem), length(ListaOutFem,TamListaFem),
        filtrar_lista(ListaPersonajes,'pelo_rubio',[],ListaOutRubio), length(ListaOutRubio,TamListaRubio),
        filtrar_lista(ListaPersonajes,'pelo_moreno',[],ListaOutMoreno), length(ListaOutMoreno,TamListaMoreno),
        filtrar_lista(ListaPersonajes,'ropa_verde',[],ListaOutRopaVerde), length(ListaOutRopaVerde,TamListaRopaVerde),
        filtrar_lista(ListaPersonajes,'ropa_roja',[],ListaOutRopaRoja), length(ListaOutRopaRoja,TamListaRopaRoja),
        filtrar_lista(ListaPersonajes,'feliz',[],ListaOutFeliz), length(ListaOutFeliz,TamListaFeliz),
        filtrar_lista(ListaPersonajes,'triste',[],ListaOutTriste), length(ListaOutTriste,TamListaTriste),
        filtrar_lista(ListaPersonajes,'joven',[],ListaOutJoven), length(ListaOutJoven,TamListaJoven),
        filtrar_lista(ListaPersonajes,'anciano',[],ListaOutAnciano), length(ListaOutAnciano,TamListaAnciano),
        filtrar_lista(ListaPersonajes,'ojos_azules',[],ListaOutAzules), length(ListaOutAzules,TamListaAzules),
        filtrar_lista(ListaPersonajes,'ojos_marrones',[],ListaOutMarrones), length(ListaOutMarrones,TamListaMarrones),
        filtrar_lista(ListaPersonajes,'con_sombrero',[],ListaOutConSombrero), length(ListaOutConSombrero,TamListaConSombrero),
        filtrar_lista(ListaPersonajes,'sin_sombrero',[],ListaOutSinSombrero), length(ListaOutSinSombrero,TamListaSinSombrero),

        seleccionar_ganador(TamListaMaso,TamListaFem,'es_chico','es_chica',GanadorSexo),
        seleccionar_ganador(TamListaRubio,TamListaMoreno,'pelo_rubio','pelo_negro',GanadorPelo),
        seleccionar_ganador(TamListaRopaVerde,TamListaRopaRoja,'ropa_verde','ropa_roja',GanadorRopa),
        seleccionar_ganador(TamListaFeliz,TamListaTriste,'feliz','triste',GanadorEstado),
        seleccionar_ganador(TamListaJoven,TamListaAnciano,'joven','anciano',GanadorEdad),
        seleccionar_ganador(TamListaAzules,TamListaMarrones,'ojos_azules','ojos_marrones',GanadorOjos),
        seleccionar_ganador(TamListaConSombrero,TamListaSinSombrero,'con_sombrero','sin_sombrero',GanadorSombrero),
        append([],[GanadorSexo,GanadorPelo,GanadorRopa,GanadorEstado,GanadorEdad,GanadorOjos,GanadorSombrero,'barba','gafas'],ListaPreguntasDisponiblesOut).
```

- Seleccionar_ganador:

```
seleccionar_ganador(TamLista1,TamLista2,Car1,Car2,Ganador):- TamLista1 =< TamLista2 -> Ganador = Car1;
                                                             Ganador = Car2.
```

OTROS ASPECTOS

En cuanto al reparto de tareas, ambos miembros del grupo hemos realizado tareas en común, relacionadas con funciones genéricas del juego, además de la forma en la que se van a almacenar los personajes y su listado. En cuanto al reparto de tareas, Robert se ha especificado mas en realizar el modo de juego entre jugadores, mientras que David se ha encargado del modo de juego contra la máquina. También es cierto, que para el modo de juego avanzado de la maquina hemos realizado un consenso entre los dos y hemos pensado la mejor estrategia posible y como adaptarla dentro de nuestro código.

En referencia al grado de cumplimiento de la práctica con respecto a los requisitos mencionados en el enunciado, se han cumplido todos los requisitos mencionados en la misma, e incluso se han realizado algunas mejoras que comentaremos a continuación. La primera de ellas es la mejora de la interfaz de usuario, donde se ha incluido un elemento del tipo 'Ascii Art'. La segunda mejora realizada es el aumento de la base de conocimiento del programa, añadiendo nuevos personajes, siendo la cantidad total de los mismos 30.

BIBLIOGRAFIA

- https://github.com/cdglasz/GuessWho
- https://github.com/anludu/GuessWho
- http://www.geekyhobbies.com/how-to-win-guess-who-within-six-turns/
- https://es.stackoverflow.com/