PROGRAMACIÓN DECLARATIVA:

Lógica y restricciones

García Martín, Ignacio

12m071

ÍNDICE

1.- Introducción

Enunciado

Supongamos el siguiente problema: a una hospedería llegan 10 peregrinos, 5 hombres y 5 mujeres, pero la hospedería solo tiene 5 habitaciones disponibles. Se acuerda echar a suertes quiénes se quedarán con las habitaciones, mediante el siguiente procedimiento: los peregrinos se disponen en círculo, uno de ellos dice un número y otro cuenta ese número de peregrinos alrededor del círculo en sentido de las agujas del reloj empezando por sí mismo; el que hace el número indicado queda fuera y se vuelve a contar a partir del siguiente. Se repite cinco veces y los cinco peregrinos que queden en el círculo se hospedarán en las habitaciones disponibles. Dada una disposición particular de los peregrinos en el círculo, ¿qué número hay que contar y por quién empezar para que las habitaciones se las queden las cinco mujeres?

En dicho problema, la disposición de los peregrinos en círculo se puede representar mediante una secuencia que empieza en uno cualquiera de ellos, continua en el orden del sentido de las agujas del reloj y termina en el adyacente al primero del lado contrario al sentido de las agujas del reloj. Si los 10 peregrinos se nombran de "a" a "j", el círculo formado por ellos en orden alfabético en el sentido de las agujas del reloj se puede representar por la secuencia "a,b,c,d,e,f,g,h,i,j" de forma que el siguiente a "j" es obviamente "a". (Nótese que también se puede representar por "b,c,d,e,f,g,h,i,j,a" y por cualquier otra secuencia ordenada alfabéticamente en la que "a" sigue a "j").

Si "m" representa una mujer y "h" un hombre, cuando la disposición del círculo es la de la secuencia "m,m,h,m,h,h,m,h,m,h" la solución al problema original es contar 29 empezando por el hombre del final (posición 10 en la secuencia). En este caso también hay solución si se quiere dejar en el círculo solo a los hombres: contar 11 empezando por la segunda mujer (posición 2). En el primer caso la secuencia final es "m,m,m,m,m" (todas las mujeres) y en el segundo caso es "h,h,h,h,h" (todos los hombres).

El problema planteado es un ejemplo particular del problema del sorteo en círculo. En esta práctica se amplía al caso general, con cualquier disposición de los círculos inicial y final.

En esta práctica se pide programar un predicado **sortear/5** tal que sortear(Ini,C,N,P,Fin) es cierto si mediante el procedimiento de sorteo explicado anteriormente se obtiene la disposición del círculo Fin a partir de la disposición del círculo inicial Ini contando N empezando en la posición P en el círculo inicial. C es una cota superior de N; por lo que no se deben considerar valores de N mayores que C. Es suficiente con que el predicado funcione cuando tanto las listas Ini y Fin como la cota C están dadas en la llamada. La ejecución debe dar como primera solución (si la hay) aquella en que hay que contar el menor número N. El programa no debe embuclarse ni al pedir más soluciones ni cuando no haya solución.

Ini y Fin son listas que representan el círculo de participantes en el sorteo tal como se ha explicado anteriormente para el problema particular de los peregrinos.

El predicado pedido debe ser general, es decir, las listas Ini y Fin pueden tener cualquier longitud y sus elementos pueden ser cualesquiera. La lista Fin debe respetar el orden de Ini, es decir, es el resultado de suprimir de Ini aquellos elementos que deben quedar fuera del círculo durante el sorteo. Por ejemplo, si Ini es [a,b,c,d] y deben quedar solo b y c entonces Fin es [b,c] y no [c,b]. Dicho de otro modo, si Ini es [a,b,c,d] y Fin es [c,b] el programa no tiene por qué dar solución.

Nota: Se puede simplificar la cuenta teniendo presente que cuando se quiere contar N en un círculo de L posiciones, siendo N>L, basta contar "N mod L".

2.- Código

Para la realización de la práctica h

Funciones Auxiliares

El predicado numberOK/3 contiene tres argumentos enteros. Este predicado nos ayuda a saber la posición de la lista a eliminar. Sea N el número que tenemos que contar, L la longitud de la lista. Si N y L coinciden, Nok es similar. En caso contrario Nok sería el modulo de N en L, con la pequeña modificación que si es cero, devolvemos L. Por ejemplo: numberOK(Nok,5,5), Nok=5

```
numberOK(L,0,L).
numberOK(Nok,N,L):-
    N > 0,
    N \== L,
    Nok is mod(N,L).
numberOK(N,N,N):- N \== 0.
```

El predicado elimina/3 tiene tres argumentos: un entero (Pos) y dos listas. Vamos avanzando en la lista y restando uno a Pos hasta situarnos en la posición Pos y devuelve la segunda lista sin el elemento.

```
elimina(1,[_|X],X).
elimina(Pos,[Ini|Inis],[Ini|Inis1]):-
    Pos1 is Pos-1,
    elimina(Pos1,Inis,Inis1).
```

El predicado nextpos/3 contiene tres argumentos enteros. Nok referido a la posición de la lista que acabamos de eliminar, LIni con la longitud de la lista de la que acabamos de eliminar el elemento y P1 la posición en la que comenzamos a contar en la lista resultante. Sabiendo que si Nok es el último elemento de la lista, debemos comenzar contando por el primer elemento de la lista resultante, en caso contrario comenzaremos a contar por la posición que indique Nok.

```
nextpos(Nok,Nok,1):- Nok \== 1.
nextpos(Nok,LIni,P1):-
    Nok \== LIni,
    P1 is Nok.
```

El predicado contar/3 contiene cinco argumentos: dos listas, una inicial y otra final; y tres enteros, el número de posiciones que contar (N), la posición desde la que comenzamos a contar(P) y las eliminaciones que tenemos que hacer (M). Sabiendo que el caso base será cuando la lista Inicial y Final sean similares cuando no queden elementos por eliminar. Para la implementación de este predicado nos hemos apoyado en los explicados anteriormente, es decir, numberOK/3, elimina/3 y nextpos/3. El funcionamiento de este predicado está planteado para que nos devuelva el número que tenemos que devolver y la posición desde la cual empezamos a contar. En primer lugar obtenemos la longitud de la lista Inicial, gracias al predicado length/2 que corresponde al valor de LOIni. Con este valor y con numberOK/3 podemos saber la posición correspondiente que obtendríamos si comenzaramos a contar en la primera posición. Sumando al resultado (Nb) la posición (P) y restando uno para sincronizar la posición con las restricciones del enunciado, podemos obtener la posición a eliminar después de aplicar de nuevo numberOK/3 para obtener la posición correcta(Nok). Obtenemos la lista (L1) que resulta de eliminar la posición (Nok) de la lista Ini descrito anteriormente y calculamos la posición (P1) por la que comenzaremos a contar en la siguiente iteración. Por último ajustamos M para saber los elementos que nos faltan por eliminar y llamamos a la misma función con L1 como lista inicial, con la misma lista Fin de objetivo, el mismo número N para contar posiciones, la posición por la que comenzar y las eliminaciones que nos faltan por hacer.

```
contar(X,X,_,_,0).
contar(Ini,Fin,N,P,M):-
    length(Ini,LOIni),
    numberOK(Nb,N,LOIni),
    Elem is Nb+P-1,
    numberOK(Nok,Elem,LOIni),
    elimina(Nok,Ini,L1),
    nextpos(Nok,LOIni,P1),
    M1 is M-1,
    contar(L1,Fin,N,P1,M1).
```

Por último, hemos definido un predicado pasos/3 con tres argumentos: dos listas y un entero. Que nos devuelve los pasos que tenemos que dar mediante la diferencia de longitudes de las listas

```
pasos(Ini,Fin,X):-
    length(Ini,LongIni),
    length(Fin,LongFin),
    X is LongIni - LongFin.
```

Sortea

El predicado exigido sortea/5 contiene 5 argumentos: Una lista Inicial de participantes(Ini) y una lista de ganadores(Fin); Una cota superior de posiciones que podemos contar (C), representado con un entero; El número de posiciones que tenemos que contar hasta obtener nuestros ganadores, siendo el menor posible. Y un entero que representa la posición por la que tenemos que empezar a contar correspondiendo la primera posición al primer concursante de la lista Inicial.

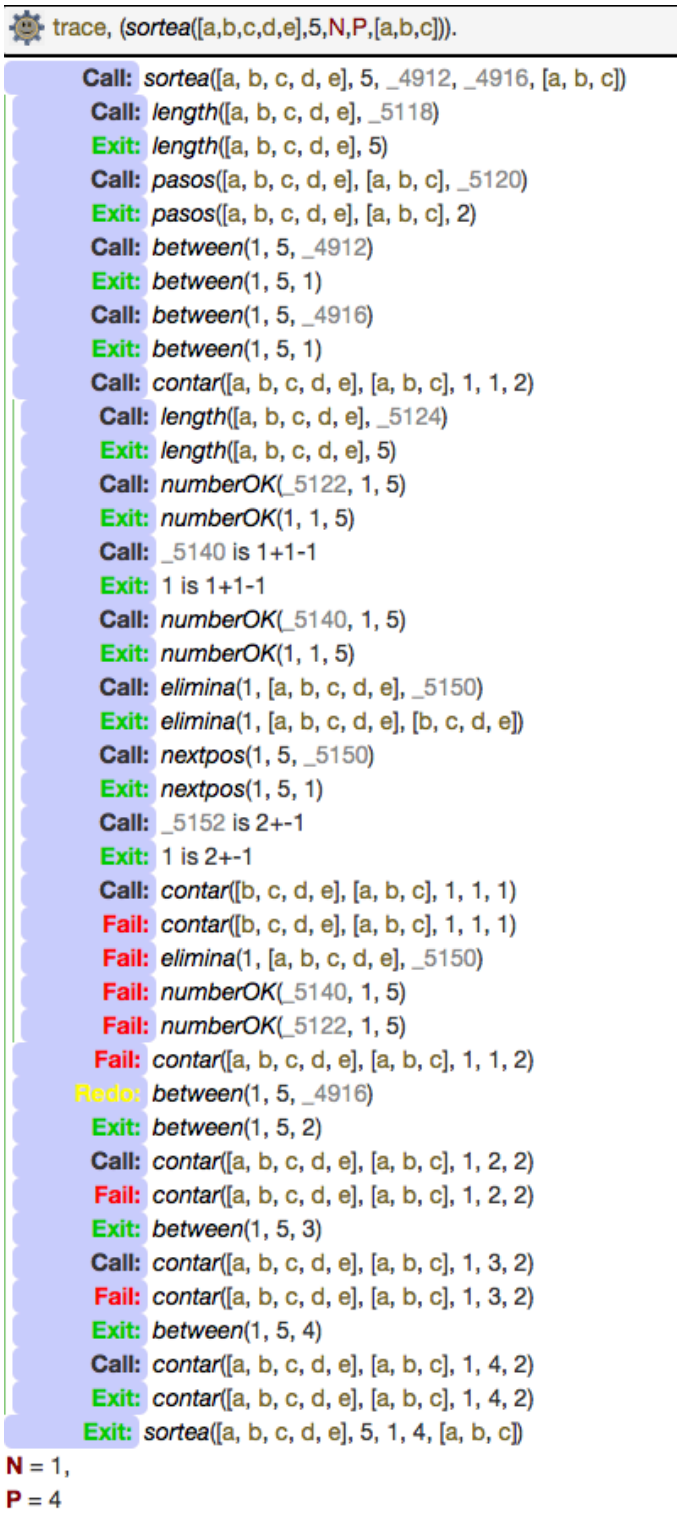
Suponiendo que la lista inicial y final sean la misma devolveremos que no tenemos que contar, es decir, devolveremos N=0.
En caso contrario obtenemos las eliminaciones que tenemos que hacer con el predicado auxiliar pasos/3. Definimos N como un valor entero entre uno y “C” con la ayuda del predicado betweeen/3. Y sabemos que P será un valor entre la primera posición de la lista y el número de concursantes que tengamos en la lista inicial, que corresponda con la longitud de esta lista. Y llamamos a nuestro predicado auxiliar N que nos devolverá el menor N posible para cumplir con nuestro objetivo desde la posición P.

```
sortea(Fin,_,0,_,Fin).
sortea(Ini,C,N,P,Fin):-
length(Ini,LongIni),
    pasos(Ini,Fin,M),
    between(1,C,N),
    between(1,LongIni,P),
    contar(Ini,Fin,N,P,M).
```

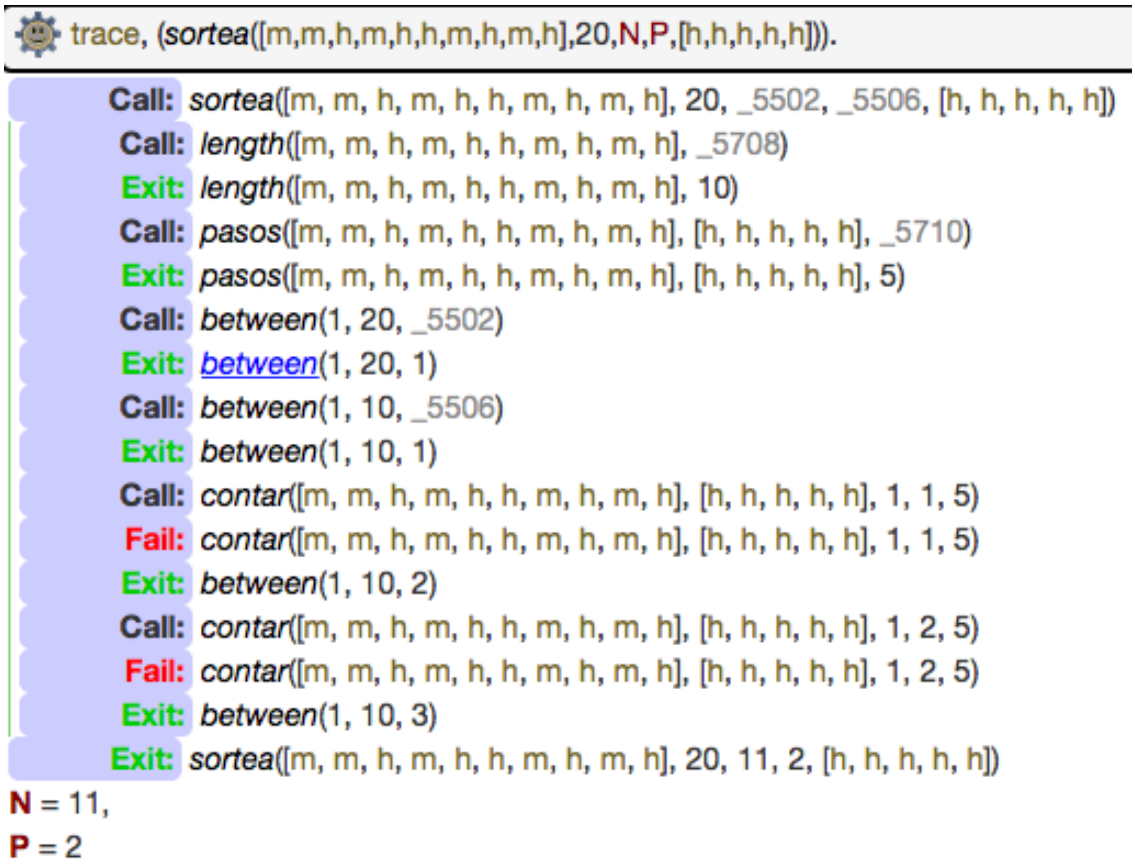
3.- Pruebas de código

Pruebas correctas

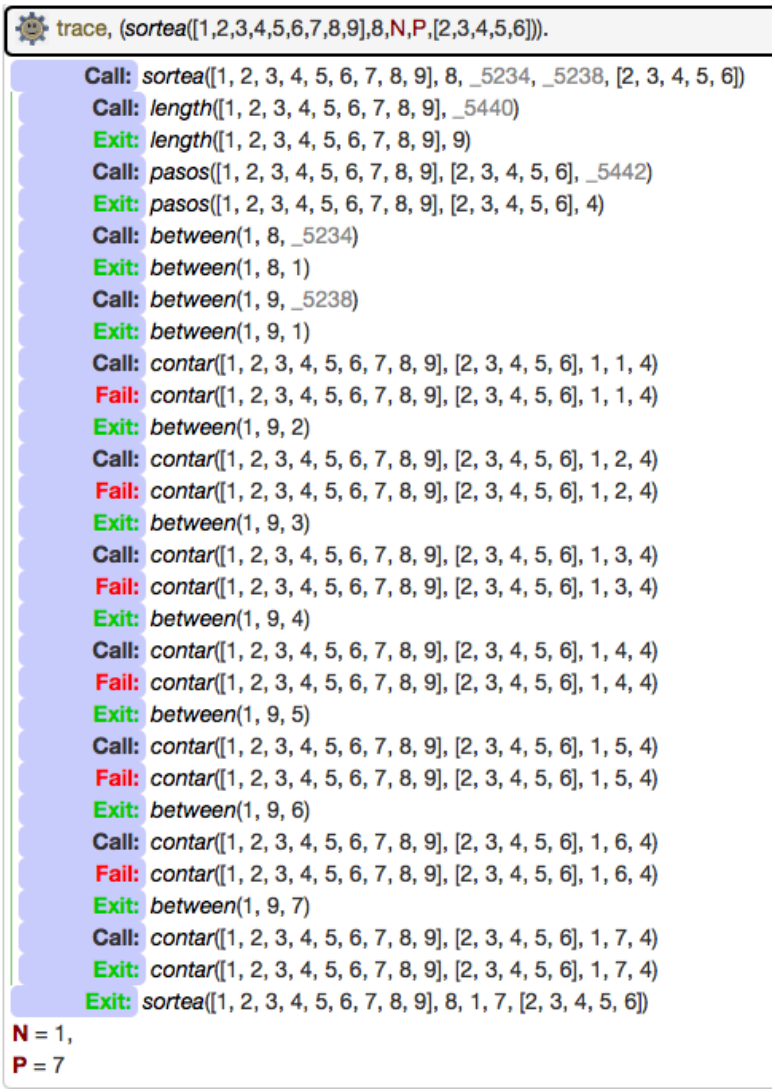
En primer lugar se presenta un caso general, correspondiente a cinco participantes de los que solo los tres primeros saldrán vencedores. Obteniendo empezando a contar en la cuarta posición de 1 en 1 obtenemos a los ganadores. Gracias al debug del código podemos ir siguiendo la traza sin problema y comprobar la veracidad de la solución.



Seguimos comprobando un ejemplo que se propone en el enunciado y obtenemos el resultado esperado. En la traza podemos ver cómo va dando errores para los diferentes valores de N. He decidido no hacer la traza completa debido a la longitud de la misma, obteniendo el resultado final una vez hemos comprobado que para P=2 n hay solución con N=1.

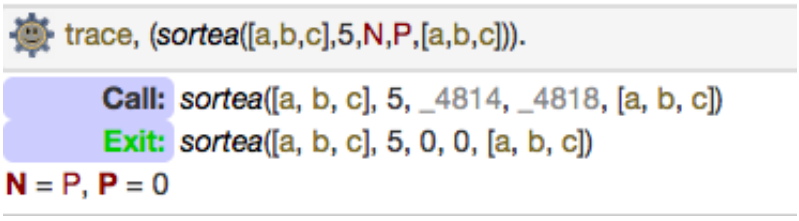


La ultima prueba correcta corresponde a la utilización de números en la lista.

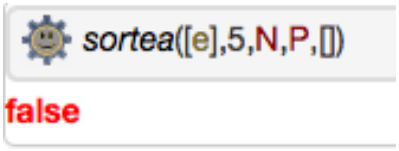


Pruebas erróneas

Se muestra a continuación la traza del programa cuando la lista inicial y final son similares, devolviendo la ausencia de número que contar y posición por la que empezar.



Ahora comprobamos cuando la lista final es una lista vacía. Es decir, no querer que nadie resulte ganador.



Por último mostraremos un ejemplo en el que la lista final contiene elementos que no se encuentran en la lista incial.