Reglas relacionadas con la historia:

- Verificar si una persona es maestro:

- Encontrar las personas que pueden defender el reino:

- Obtener las personas que están ayudando en la base de conocimiento de la historia:

- Domina una habilidad:

- Verificar si una persona es aprendiz:

Consultas con variables libre e instanciadas:

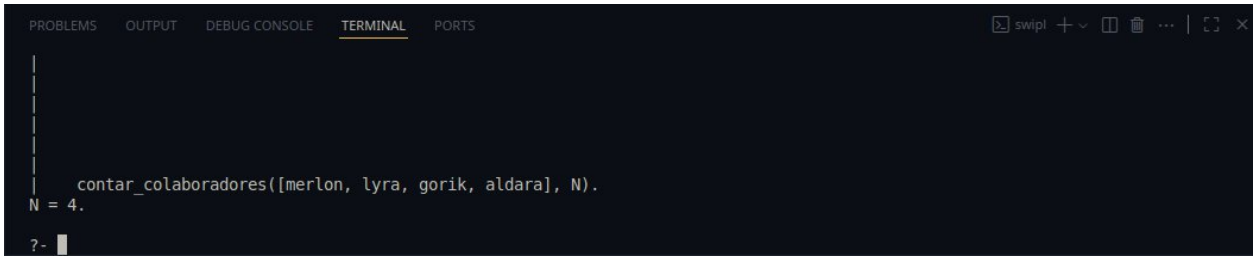
- Buscamos personas que sean magos:

- Buscamos personas que sepa sobre estrategia:

- Comprobamos si Aldara esta estudiando con Merlon:

Regla recursiva:

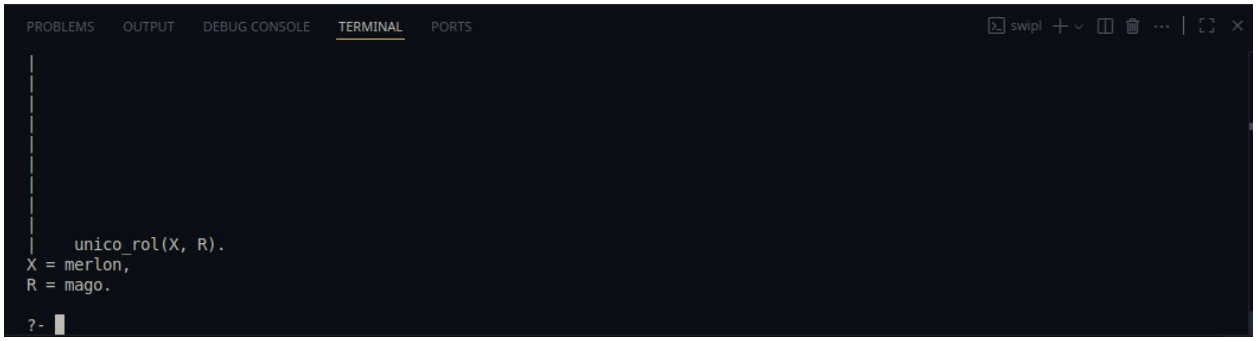
- Obtenemos el numero de elementos de una lista, en este caso los colaboradores de la base de conocimiento de la historia:



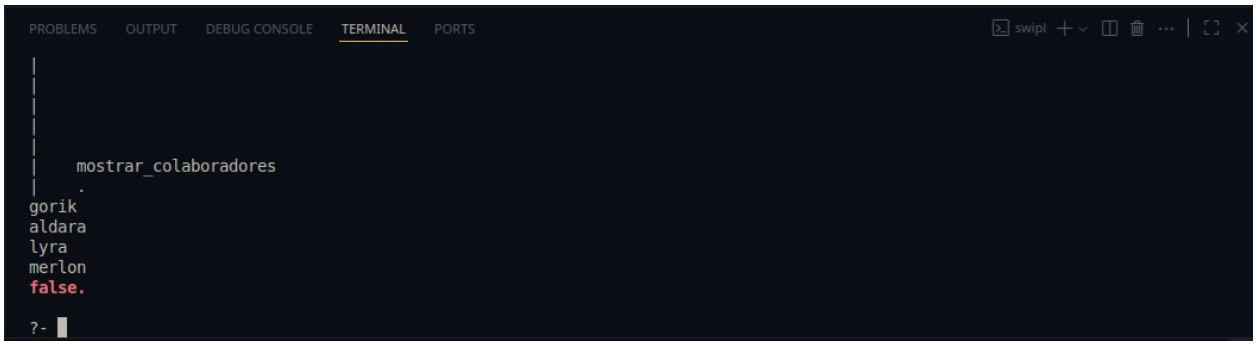
Predicados usando \+, ! y fail.

- Verificamos que una persona no sea un guerrero:

- Obtenemos un rol y luego cortamos la ejecución.



- Mostramos todos los colaboradores forzando el backtraking hasta quedarse sin opciones:



Preguntas de análisis y reflexión:

1. ¿Qué elementos componen tu base de conocimientos (hechos, relaciones y reglas)?
    a. Hechos: la profesión de las diferentes personas que aparecen en la historia y los conocimientos que cada uno sabe.
    b. Relaciones: las amistades que tienen los personajes, las relaciones de maestro-aprendiz y por ultimo como los objetos se relacionan como diferentes acciones como proteger el reino o ayuda en la construcción de la base de conocimientos.
    c. Reglas: ayudan a modelar diferentes relaciones para extraer conocimientos derivados que no se pueden ver a simple vista. Sabe que personaje son maestros, que personajes son responsables de defender el reino, quienes están ayudando a construir la base de conocimiento, que habilidad poseen los diferentes personajes, conocer los personajes que son aprendices y listar todos los personajes que ayudan en la creación de la base de conocimiento de forma recursiva.
2. Escribe una regla en forma de Cláusula de Horn y explica su propósito.
   es_maestro(X) :- enseña(X, _), sabe(X, _): su propósito es validar que una persona sea un verdadero maestro, no solo sabiendo si esta a cargo de un aprendiz, sino que también validamos que tenga conocimientos en alguna área para compartir con los aprendices.
3. Muestra una consulta con variable libre y otra con variable ligada.
    a. Variable libre:

   Aquí la variable libre nos sirve para encontrar una persona que tenga el rol de mago, prolog busca en la base los diferentes hechos hasta dar con el indicado y lo liga a la variable X.
    b. Variable ligada:

   En este caso la variables esta ligadas, cada una posee un valor definido antes

de consultarlo, en este caso le decimos a prolog si Merlon es maestro de Aldara, prolog busca en la base si esto es verdad, si lo es devuelve true de lo contrario false.

4. Representa o describe el árbol de resolución de una de tus consultas.
   es(X, mago):
   a. Prolog busca hechos con la estructura de es(alguien, mago)
   b. Intentea unificar es(X, mago) con cada hecho que encontro anteriormente:
      i. es(merlon, mago) -> si, hay un hecho que coincide.
      ii. es(aldara, aprendiz) -> no, no coincide
      iii. es(gorik, guerrero) -> no, no coincide
      iv. es(lyra, bibliotecaria) -> no, no coincide
   c. Prolog encuentra una solución y pasa a realizar un proceso de unificación y liga el valor correspondiente a la variable libre X.
5. Explica el caso trivial y el caso general del ejemplo recursivo que desarrollaste.
   a. Caso base:
   ```
   %caso base
   contar_colaboradores([], 0).
   ```
   Básicamente nos dice que si la lista esta vacía el numero de colaboradores es 0, ya que en recursión necesitamos una condición de paro.
   b. Caso trivial:
   ```
   %caso trivial
   contar_colaboradores([R|Resto], N) :-
      contar_colaboradores(Resto, N1),
      N is N1 + 1.
   ```
   Tomas una lista de al menos un elemento, tomamos el primero de la lista y sumamos 1 a N, luego se llama recursivamente con el resto de la lista y se vuelve a repetir la recusión tomando el primer elemento y sumando de nuevo 1, asi hasta obtener vaciar la lista, ahora sumamos en reversa el resultado para obtener el total.
6. Describe cómo aplicaste los predicados \+, ! y fail y cuál fue su efecto.
   a. \+: responde verdadero si X no es un guerrero, el operador \+ niega la consulta, así que solo será cierto para quienes no tengan el rol de guerrero.
   b. !: detiene el backtracking una vez que encuentra una combinación de X y R que cumple es(X, R), así se asegura que solo se devuelva el primer rol encontrado.
   c. fail: fuerza a Prolog a buscar todas las soluciones posibles, imprime todos los colaboradores que ayudan en base_de_conocimiento y al final la consulta falla debido a que se queda sin mas opciones.
7. Reflexiona: ¿de qué manera tu base de conocimiento puede considerarse un sistema inteligente lógico?
   Debido a que posee reglas que moldean los hechos y las relaciones descritas en la base, de esta manera podemos no solo consultar las relaciones que tiene los hechos, con la ayuda de las reglas podemos simplificar la información o automatizarla para generar respuestas mas complejas a lo descrito en la base de conocimiento. Ademas podemos agregar nuevos hechos en la base sin modificar las reglas ya escritas, de esta

manera nuestra capacidad de generar nueva información se amplia sin la necesidad de realizar grandes cambios.