Guia de Trabajos Prácticos
Paradigma Lógico

Parte I

Introducción

Esta guía está diseñada para introducir a los estudiantes en el paradigma de la programación lógica utilizando Prolog. Los ejercicios proporcionados ayudarán a desarrollar habilidades en la creación y consulta de cuerpos de conocimiento, la construcción de árboles de resolución, y la modelización de relaciones complejas mediante predicados. Cada ejercicio incluye un conjunto de instrucciones detalladas para guiar a los estudiantes a través del proceso de implementación y verificación de sus programas.

Ejercicio 1: Consultas en un Cuerpo de Conocimiento

A continuación, se presenta un cuerpo de conocimiento en Prolog. Responda qué retornarán las consultas dadas este cuerpo de conocimiento.

Cuerpo de Conocimiento

```
f(a, 2).
f(a, 3).
f(b, 2).
f(b, 4).
f(c, 1).
f(c, 2).
```

Consultas

a) f(X, 1).
b) f(X).
c) f(a, X).
d) f(c, 1).
e) f(X, Y).
f) f(2, a).
g) f(X, Y), f(X, 4).

Instrucciones

1. Lea atentamente cada consulta y trate de predecir el resultado basándose en el cuerpo de conocimiento.

2. Ejecute las consultas en un entorno de Prolog para verificar sus respuestas.
3. Compare sus predicciones con los resultados obtenidos y explique cualquier discrepancia.

Ejercicio 2: Creación y Consulta de un Cuerpo de Conocimiento en Prolog

Escriba como programa en Prolog el siguiente cuerpo de conocimientos:

- Leoncio es padre de Alberto y de Gerónimo.
- Alberto es padre de Juan y de Luis.
- Gerónimo es padre de Luisa.
- A es hermano de B si A y B tienen un padre en común y son distintas personas.
- A es nieto de B si el padre de A es hijo de B.

Cuerpo de Conocimiento

```
padre(leoncio, alberto).
padre(leoncio, geronimo).
padre(alberto, juan).
padre(alberto, luis).
padre(geronimo, luisa).
hermano(A, B) :- padre(P, A), padre(P, B), A \= B.
nieto(A, B) :- padre(A, P), padre(P, B).
```

Consultas

a) ¿Cómo consultaría si Alberto es padre de Luis? padre(alberto,luis)
b) ¿Cómo consultaría si Luis es padre de Alberto? padre(luis,alberto)
c) ¿Cómo consultaría quién es hermano de Luis? hermano(X,luis) y hermano(luis,X)
d) ¿Cómo consultaría de quién es nieto Luisa? nieto(X,luisa)
e) ¿Cómo consultaría quién es nieto de quién? nieto(X,Y)

Instrucciones

1. Escriba el cuerpo de conocimiento en un archivo Prolog.
2. Ejecute las consultas en un entorno de Prolog para verificar sus respuestas.
3. Compare sus predicciones con los resultados obtenidos y explique cualquier discrepancia.

Ejercicio 3: Declaración y Consulta de Menús en Prolog

Dada la lista de platos de un restaurante, confeccione los predicados necesarios para declarar los menús, y luego realice las siguientes consultas:

Menús Existentes

```
menu(['Bombones de jamon', 'Locro', 'Dulce de batata']).
menu(['Bombones de jamon', 'Locro', 'Alfajor norteño']).
menu(['Tarta de Atun', 'Atados de repollo', 'Dulce de batata']).
menu(['Tarta de Atun', 'Pollo romano con hierbas y vino', 'Flan']).
menu(['Volovanes de atun', 'Matambre con espinacas y parmesano', 'Torta moka']).
menu(['Buñuelos de bacalao', 'Pollo romano con hierbas y vino', 'Alfajor norteño']).
```

Consultas

a) Listado completo de posibles menús que se ofrecen. menu(X,Y,Z).
b) ¿En qué menú está incluido el postre 'Dulce de batata'? menu(X,Y,'Dulce de batata').
c) ¿En qué menú está incluido el plato principal 'Locro'? menu(X,'Locro',Z)
d) ¿Hay algún menú que contenga como plato principal 'Pato a la naranja'? menu(X,...,Z)
e) ¿Hay algún menú que contenga 'Locro' como entrada? menu('Locro',Y,Z)

Instrucciones

1. Escriba los predicados necesarios para declarar los menús en un archivo Prolog.
2. Ejecute las consultas en un entorno de Prolog para verificar sus respuestas.
3. Compare sus predicciones con los resultados obtenidos y explique cualquier discrepancia.

Ejercicio 4: Construcción del Árbol de Resolución

Construya el árbol de resolución para la consulta del punto “c” del ejercicio 3.

Consulta del Punto “c”:

```
menu(M), member('Locro', M).
```

Instrucciones

1. Dibuje el árbol de resolución mostrando cómo Prolog intenta resolver la consulta menu(M), member('Locro', M).
2. Asegúrese de incluir todas las ramas y fallos hasta llegar a las soluciones.
3. Documente cada paso del proceso de resolución.

Ejercicio 5: Declaración y Consulta de Rutas de Colectivo en Prolog

Dadas las siguientes rutas de colectivo, confeccione los predicados necesarios para declarar las rutas y luego realice las siguientes consultas:

Rutas Existentes

```
ruta(santafe, parana).
ruta(parana, corrientes).
ruta(santafe, cordoba).
ruta(santafe, coronda).
ruta(santafe, rosario).
ruta(rosario, capital).
ruta(rosario, mardelplata).
ruta(capital, cordoba).
```

Consultas

a) ¿Desde que orígenes se llega a Córdoba?
b) ¿Qué destinos son alcanzados desde Paraná?
c) ¿Hay alguna ruta entre Paraná y Córdoba?
d) ¿Hay alguna combinación de dos rutas que permita ir desde Santa Fe a Corrientes?

Instrucciones

1. Escriba los predicados necesarios para declarar las rutas en un archivo Prolog.
2. Ejecute las consultas en un entorno de Prolog para verificar sus respuestas.
3. Compare sus predicciones con los resultados obtenidos y explique cualquier discrepancia.

Ejercicio 6: Modelado del Sistema Solar en Prolog

Cree un programa en Prolog que modele el sistema solar. Para ello se deberá utilizar dos predicados: estrella/1 y órbita/2. estrella/1 define la estrella del sistema solar y órbita/2 hace referencia a la relación entre dos cuerpos, de los cuales el primero es el orbitado y el segundo el orbitante.

Información a Modelar

- **Estrella**: sol
- **Planetas**: mercurio, venus, tierra, marte, jupiter, saturno, urano, neptuno, pluton
- **Lunas de mercurio**: no tiene.
- **Lunas de venus**: no tiene.
- **Lunas de tierra**: luna.
- **Lunas de marte**: deimos, phobos.
- **Lunas de jupiter**: adrastea, aitne, amalthea, ananke, aoede, arche, autonoe, callisto, carme, callirrhoe, carpo, chaldene, cyllene, elara, erinome, euanthe, eukelade, euporie, europa, eurydome, ganymede, harpalyke, hegemone, helike, hermippe, himalia, io, iocaste, isonone, kale, kallichore, kalyke, kore, leda, lysithea, magaclite, metis, mneme, orthosie, pasiphae, pasithee, praxidike, sinope, sponde, s2000j11, s2003j2, s2003j3, s2003j4, s2003j5, s2003j9,

s2003j10, s2003j12, s2003j15, s2003j16, s2003j17, s2003j18, s2003j19, s2003j23, taygete, thebe, thelxinoe, themisto, thyone.

- **Lunas de saturno**: aegir, albiorix, atlas, bebhionn, bergelmir, bestla, calypso, daphnis, dione, enceladus, epimetheus, erriapo, farbauti, fenrir, fornjot, hati, helene, hyperion, hyrokkin, iapetus, ijiraq, janus, kari, kiviuq, loge, methone, mimas, mundilfari, narvi, paaliaq, pallene, pan, pandora, phoebe, polydeuces, prometheus, rhea, siarnaq, skathi, skoll, surtur, suttungr, s2004s07, s2004s12, s2004s13, s2004s17, s2006s1, s2006s3, s2006s4, s2007s1, s2007s2, s2007s3, tarvos, telesto, tethys, thrymr, titan, ymir.
- **Lunas de urano**: ariel, belinda, bianca, caliban, cordelia, cressida, cupid, desdemona, juliet, mab, margaret, miranda, oberon, ophelia, portia, prospero, puck, rosalind, setebos, stephano, s1986u10, s2001u2, s2001u3, titania, trinculo, umbriel.
- **Lunas de neptuno**: despina, galatea, halimede, larissa, laomedeia, naiad, nereid, neso, proteus, psamathe, sao, thalassa, triton.
- **Lunas de pluton**: charon, nix, hydra.

Predicados a Crear

1. planeta/1: unifica si su parámetro es un planeta del sistema solar.
2. luna/1: unifica si su parámetro es una luna del sistema solar.
3. lunaDe/2: unifica si su primer parámetro es un planeta y su segundo parámetro es una luna de dicho planeta.

Ejemplo de Código

```
estrella(sol).
planeta(mercurio).
planeta(venus).
planeta(tierra).
planeta(marte).
planeta(jupiter).
planeta(saturno).
planeta(urano).
planeta(neptuno).
planeta(pluton).
luna(luna).
luna(deimos).
luna(phobos).
luna(adrastea).
% (Agregar todas las lunas de los planetas)
```

Instrucciones:

1. Escriba los predicados necesarios para modelar el sistema solar en un archivo Prolog.
2. Ejecute consultas para verificar las relaciones entre planetas, lunas y la estrella.

3. Compare sus predicciones con los resultados obtenidos y explique cualquier discrepancia.

Parte II

Introducción

En esta segunda parte de la guía, se profundizará en la creación y consulta de cuerpos de conocimiento más complejos utilizando Prolog. Los ejercicios presentados están diseñados para desafiar a los estudiantes a pensar lógicamente y a construir programas que modelen situaciones del mundo real. A través de estos ejercicios, los estudiantes aprenderán a definir relaciones y predicados en Prolog, a trabajar con árboles genealógicos, a modelar gustos literarios y a combinar elementos en menús de restaurantes. La práctica constante de estos conceptos ayudará a fortalecer la comprensión de la programación lógica y sus aplicaciones.

Ejercicio 1: Menú del Restaurante

Escriba un programa en Prolog que represente el cuerpo de conocimiento con los distintos platos de un restaurante: entrada, plato principal y postre. Cree el cuerpo de conocimiento con tres ejemplos de cada uno. Por último, cree un predicado carta/3 que como resultado combine todos los posibles platos, en donde el primer argumento corresponda a las entradas, el segundo a los platos principales y el tercero a los postres. ¿Cuántos resultados debería retornar?

Ejemplo

```
?- carta(X, Y, Z).
X = ensalada, Y = lomo a la pimienta, Z = flan;
...
```

Instrucciones

1. Defina tres predicados separados para las entradas, platos principales y postres.
2. Agregue tres ejemplos de cada tipo de plato.
3. Cree un predicado carta/3 que combine todas las entradas, platos principales y postres.
4. Realice una consulta para carta/3 y verifique que se generen todas las combinaciones posibles.

Ejercicio 2: Árbol Genealógico

Con los siguientes predicados, cree un árbol genealógico correcto con al menos 4 niveles y 15 miembros:

- `progenitor/2`: determina que el primer argumento es progenitor (madre o padre) del segundo.
- `hombre/1`: determina que su argumento es una persona de sexo masculino.
- `mujer/1`: determina que su argumento es una persona de sexo femenino.

Cree las siguientes reglas:

a) El padre de una persona es su progenitor masculino.
b) La madre de una persona es su progenitor femenino.
c) El abuelo de una persona es el padre del padre o el padre de la madre de dicha persona.
d) La abuela de una persona es la madre del padre o la madre de la madre de dicha persona.
e) Un hermano de una persona es cualquier persona con los mismos padres y que sea distinto de la persona en cuestión.
f) Un hermano varón de una persona es cualquier persona varón con los mismos padres y que sea distinto de la persona en cuestión.
g) Una hermana mujer de una persona es cualquier persona mujer con los mismos padres y que sea distinta de la persona en cuestión.
h) Un sucesor de una persona es cualquier persona que descienda de ésta.

Cree las siguientes reglas interpretando la lógica a utilizar:

a) `es_madre/1`: su argumento es madre.
b) `es_padre/1`: su argumento es padre.
c) `tia/2`: su primer argumento es tía del segundo argumento.
d) `yerno/2`: su primer argumento es yerno de su segundo argumento.
e) `nuera/2`: su primer argumento es nuera de su segundo argumento.

Instrucciones

1. Defina los predicados hombre/1, mujer/1 y progenitor/2 para cada miembro del árbol genealógico.
2. Utilice los predicados básicos para definir las reglas más complejas, como padre/2, madre/2, abuelo/2, etc.
3. Asegúrese de que las reglas para hermano/2, hermano_varon/2 y hermana_mujer/2 considerar que los hermanos deben ser diferentes entre sí.
4. Realice consultas para verificar que las relaciones genealógicas se hayan definido correctamente.

Ejercicio 3: Gustos Literarios

Construir un programa Prolog que pueda responder consultas acerca de los gustos literarios de distintos grupos de personas, a partir de la siguiente información:

- A los abogados les gustan las novelas largas.
- Tanto a los ingenieros como a los médicos les gustan las novelas.
- A las mujeres les gustan todos los libros largos.
- A los contadores varones les gustan tanto los libros de cuentos como los libros de poemas.
- Haydée es una mujer que es abogada y también ingeniera.
- Tania es una mujer médica.
- Livio es un varón contador a quien le gusta "Rayuela".
- Pedro es un abogado varón a quien le gustan los libros de cuentos.
- "Rayuela" y "Karamazov" son novelas largas.
- "Octaedro" es un libro de cuentos corto.
- "Inventario" es un libro de poemas largo.
- "Leones" es una novela corta.

Consultas

a) ¿Qué libros le gustan a Livio?
b) ¿A quiénes les gusta "Leones"?
c) ¿Qué libros cortos tiene registrado el programa?

Agregar al programa la posibilidad de responder consultas acerca de qué libros son valiosos. Se considera que un libro es valioso si le gusta a, al menos, dos personas distintas.

Instrucciones

1. Defina los predicados que caracterizan los libros (libro/3) y los gustos de las personas (gusta/2).
2. Utilice los predicados de profesión para determinar los gustos literarios.
3. Asegúrese de considerar tanto las características del libro como las preferencias de las personas para definir los predicados gusta/2.
4. Para responder consultas sobre libros valiosos, cuente cuántas personas diferentes gustan de cada libro.
5. Realice consultas para verificar que el programa responda correctamente sobre los gustos literarios y los libros valiosos.

Parte III

Introducción

En esta guía, exploraremos conceptos avanzados de la programación lógica utilizando Prolog. Los ejercicios se centran en el uso de listas, recursión, árboles binarios y operaciones comunes en

estructuras de datos. Cada ejercicio está diseñado para profundizar en la lógica y la manipulación de datos en Prolog, ofreciendo una comprensión más sólida de cómo resolver problemas complejos de manera lógica.

Ejercicio 1: Calcular el Factorial de un Número

Cree un programa en Prolog que permita calcular el factorial de un número con el predicado factorial/2, validando que dicho número sea mayor o igual a cero.

Ejemplo

```
?- factorial(5, Factorial).
Factorial = 120.
```

Instrucciones

1. Defina el predicado factorial/2.
2. Asegúrese de validar que el número sea mayor o igual a cero.
3. Implemente la lógica para calcular el factorial de un número.

Ejercicio 2: Contar Apariciones en una Lista

Cree un programa en Prolog que cuente la cantidad de veces que aparece un elemento en una lista.

Ejemplos

```
?- contar(y, [a, b, c, a, d, e, a, f, a], Cantidad).
Cantidad = 0.

?- contar(a, [a, b, c, a, d, e, a, f, a], Cantidad).
Cantidad = 4.
```

Instrucciones

1. Defina el predicado contar/3.
2. Implemente la lógica para contar las apariciones de un elemento en una lista.

Ejercicio 3: Contar Elementos de una Lista

Escriba un programa en Prolog que reciba como primer parámetro una lista de números y unifique el segundo con la cantidad de elementos de dicha lista.

Ejemplo

```
?- cantidad([a, b, c], Elementos).
Elementos = 3.
```

Instrucciones

1. Defina el predicado cantidad/2.
2. Implemente la lógica para contar los elementos de una lista.

Ejercicio 4: Sumar Números en una Lista

Escriba un programa en Prolog que, dada una lista de números enteros, calcule el resultado de sumar dichos números.

Ejemplo

```
?- suma([1, 2, 3], X).
X = 6.
```

Instrucciones

1. Defina el predicado suma/2.
2. Implemente la lógica para sumar los números de una lista.

Ejercicio 5: Filtrar Números Positivos

Escriba un programa en Prolog que, dada una lista de números enteros, retorne otra lista solo con los números positivos de la misma.

Ejemplo

```
?- positivos([1, -2, 3, -4], ListaPositivos).
ListaPositivos = [1, 3].
```

Instrucciones

1. Defina el predicado positivos/2.
2. Implemente la lógica para filtrar los números positivos de una lista.

Ejercicio 6: Sumar Elemento a Elemento de Dos Listas

Escriba un programa en Prolog que reciba dos listas de números, verifique que sean de la misma longitud, y luego retorne una lista con la suma elemento a elemento de ambas listas.

Ejemplo

```
?- suma_lista([1, -2, 3, -4], [2, 3, 1, 4], ListaSuma).
ListaSuma = [3, 1, 4, 0].
```

Instrucciones

1. Defina el predicado suma_lista/3.
2. Implemente la lógica para sumar elemento a elemento de dos listas de la misma longitud.

Ejercicio 7: Eliminar Duplicados de una Lista

Escriba un programa en Prolog que, dada una lista, elimine todos los elementos duplicados de la misma.

Ejemplo

```
?- eliminar_dup([1, 2, 3, 1, 4, 3, 5, 6], SinDup).
SinDup = [1, 2, 3, 4, 5, 6].
```

Instrucciones:

1. Defina el predicado eliminar_dup/2.
2. Implemente la lógica para eliminar duplicados de una lista.

Ejercicio 8: Determinar la Profundidad de un Árbol Binario

Escriba un programa en Prolog que recorra un árbol binario y determine la profundidad del mismo. La representación del árbol será una lista con el siguiente formato: [I, N, D].

Ejemplo

```
?- profundidad([[[c], b, [d]], a, [[], e, [f]]], Profundidad).
Profundidad = 3.
```

Instrucciones

1. Defina el predicado profundidad/2.
2. Implemente la lógica para determinar la profundidad de un árbol binario representado como listas.

Ejercicio 9: Insertar Elemento en una Lista Ordenada

Escriba un programa en Prolog que, dada una lista numérica ordenada, inserte un elemento en el lugar correspondiente según el orden.

Ejemplo

```
?- insertar(3, [1, 2, 4, 5], Resultado).
Resultado = [1, 2, 3, 4, 5].
```

Instrucciones:

1. Defina el predicado insertar/3.
2. Implemente la lógica para insertar un elemento en una lista ordenada.

Ejercicio 10: Ordenar una Lista de Números

Escriba un programa en Prolog que recursivamente ordene una lista de números enteros.

Ejemplo

```
?- ordenar([2, 4, 3, 1], ListaOrdenada).
ListaOrdenada = [1, 2, 3, 4].
```

Instrucciones

1. Defina el predicado ordenar/2.
2. Implemente la lógica para ordenar una lista de números enteros de manera recursiva.

Ejercicio 11: Aplanar una Lista

Escriba un programa en Prolog que aplane una lista. El predicado aplanar/2 recibe una lista cuyos elementos pueden ser otras listas y debe retornar una lista con todos los elementos atómicos presentes.

Ejemplos

```
?- aplanar([1, 2, 3], ListaPlana).
ListaPlana = [1, 2, 3].
?- aplanar([1, 2, [3]], ListaPlana).
ListaPlana = [1, 2, 3].
?- aplanar([1, [2, [3]]], ListaPlana).
ListaPlana = [1, 2, 3].
```

Instrucciones:

1. Defina el predicado aplanar/2.
2. Implemente la lógica para aplanar una lista, manejando elementos que pueden ser otras listas.

Ejercicio 12: Permutaciones de una Lista

El siguiente programa en Prolog calcula las permutaciones de los elementos de una lista.

Código

```
ins(X, L, [X | L]).
ins(X, [Y | L1], [Y | L2]) :- ins(X, L1, L2).
per([], []).
per([X | L], Lp) :- per(L, L1), ins(X, L1, Lp).
```

Ejercicios

a) Ejecute el programa y escriba el resultado obtenido para per([1, 2, 3], L).

b) Explique en sus propios términos cuál es la lógica que utiliza el programa para obtener las permutaciones.

Instrucciones

1. Ejecutar el programa en un entorno de Prolog.
2. Analice y explique la lógica del programa.

Parte IV

Introducción

En esta guía, se explorará el uso avanzado del operador de corte en Prolog para optimizar programas y evitar unificaciones innecesarias. El operador de corte (!) permite controlar el flujo de ejecución y es útil para mejorar la eficiencia de los predicados. Los ejercicios presentados cubren temas como el cálculo de factoriales, manipulación de listas, y la evaluación de predicados con operadores de corte.

Ejercicio 1: Factorial de un Número con Operador de Corte

Dado el siguiente programa en Prolog que calcula el factorial de un número:

```
factorial(0, 1).
factorial(Numero, Factorial) :-
    Numero > 0,
    NumeroAnt is Numero - 1,
    factorial(NumeroAnt, FactorialAnt),
    Factorial is Numero * FactorialAnt.
```

El resultado de evaluar factorial(5, X). es:

```
X = 120;
false.
```

Utilice un operador de corte donde corresponda para que el programa finalice una vez terminada la recursión y no retorne el false final.

Instrucciones

1. Añada el operador de corte (!) al predicado factorial/2 para evitar que el programa siga buscando otras soluciones después de encontrar la primera.

Ejercicio 2: Evaluación de Predicados con Operador de Corte

Dado el siguiente programa en Prolog:

```
p(1).
p(2):- !.
p(3).
```

Evaluar las siguientes consultas, dejando registrado para cada una el resultado obtenido y explicando en lenguaje natural por qué se obtiene dicho resultado, detallando los puntos de elección que han sido desechados y cuál es el operador de corte por el cual han sido quitados.

Consultas

a) p(X).

b) p(X), p(Y).

c) p(X), !, p(Y).

Instrucciones

1. Ejecutar cada consulta en un entorno de Prolog.
2. Registre los resultados obtenidos.
3. Explique en lenguaje natural el comportamiento del programa en cada consulta.

Ejercicio 3: Eliminar el Primer Elemento de una Lista

Implementar un predicado eliminar_primero/3 que quite de una lista la primera aparición de un determinado elemento, utilizando el operador de corte donde considere necesario para garantizar que no se realicen unificaciones innecesarias.

Ejemplos

```
?- eliminar_primero([1, 2, 3, 1, 2, 3], 2, X).
X = [1, 3, 1, 2, 3].
```

```
?- eliminar_primero([1, 2, 3, 1, 2, 3], a, X).
X = [1, 2, 3, 1, 2, 3].
```

Instrucciones

1. Defina el predicado eliminar_primero/3.
2. Utilice el operador de corte para evitar unificaciones innecesarias.

Ejercicio 4: Agregar un Elemento Nuevo a una Lista

Implementar un predicado agregar_nuevo/3 que agregue a una lista un elemento, solo si la misma no lo contiene. Utilizar el operador de corte para no proceder en caso de detectar que el elemento ya existe.

Ejemplo:

```
?- agregar_nuevo(a, [1, 2, 3], X).
X = [1, 2, 3, a].

?- agregar_nuevo(2, [1, 2, 3], X).
X = [1, 2, 3].
```

Instrucciones

1. Defina el predicado agregar_nuevo/3.
2. Utilice el operador de corte para evitar agregar el elemento si ya existe en la lista.

Ejercicio 5: Evaluar la Semejanza entre dos Palabras

Dadas dos palabras representadas como listas de caracteres, evaluar la semejanza entre las mismas. Para esto se verificarán, por posición, las letras de las dos palabras; cada coincidencia sumará un punto y cada vez que las letras no coincidan se restará un punto.

```
?- semejanza([h,o,l,a], [h,o,l,o], S).
S = 2.

?- semejanza([m,e,s,a], [m,e,s,a,d,a], S).
S = 2.

?- semejanza([s,o,l,a], [m,o,n,a], S).
S = 0.

?- semejanza([s,o,l], [c,a,s,a], S).
S = -4.
```

Instrucciones:

1. Defina el predicado semejanza/3.
2. Implemente la lógica para evaluar la semejanza entre dos palabras, sumando y restando puntos según las coincidencias y diferencias.

Ejercicio 6: Buscar una Palabra en un Diccionario

Dado un diccionario en forma de lista de palabras, se quiere buscar una palabra cualquiera. Si la palabra pertenece al diccionario, se debe devolver una lista de un elemento con la palabra buscada. Si no existe, se debe devolver una lista con las alternativas a la palabra buscada y su valoración de semejanza.

Diccionario

```
dic([sanar, hola, sabana, sabalo, prueba, computadora, cartera, mate, termo, mesa, silla,
sarna]).
```

Ejemplos

```
?- buscar(hola, L).
L = [hola].

?- buscar(holo, L).
L = [[hola, 2]].

?- buscar(saban, L).
L = [[sanar, 1], [sabana, 4], [sabalo, 2]].
```

Instrucciones:

1. Defina el predicado buscar/2.
2. Utilice el predicado semejanza/3 para calcular la semejanza entre palabras.
3. Utilice el predicado atom_chars/2 para convertir átomos en listas de caracteres.

Ejercicio 7: Reemplazar Elementos en una Lista

Descripción

Escriba un predicado en Prolog de aridad 6 que implemente la función reemplazar, la cual recibirá los siguientes parámetros:

1. El elemento a reemplazar en la primera lista.
2. El elemento de reemplazo.
3. A partir de qué instancia encontrada comienza a reemplazar (validar que su valor sea >= 1).

4. Cuántos reemplazos como máximo se harán (siendo -1 el valor para indicar que reemplace todas las instancias que encuentre; validar que su valor sea -1 o >= 1).
5. Una lista de elementos en donde se reemplazará.
6. El último parámetro debe unificar con la lista resultante.

Ejemplos

```
?- reemplazar(1, a, 0, -1, [1, 1, 2, 2, 3, 3, 2, 2, 1, 1], L).
% El valor de inicio debe ser mayor o igual a 1
% Fail.

?- reemplazar(1, a, 1, -2, [1, 1, 2, 2, 3, 3, 2, 2, 1, 1], L).
% La cantidad de reemplazos debe ser -1 o mayor o igual 1
% Fail.

?- reemplazar(1, a, 1, -1, [1, 1, 2, 2, 3, 3, 2, 2, 1, 1], L).
% L = [a, a, 2, 2, 3, 3, 2, 2, a, a]

?- reemplazar(1, a, 1, 1, [1, 1, 2, 2, 3, 3, 2, 2, 1, 1], L).
% L = [a, 1, 2, 2, 3, 3, 2, 2, 1, 1]

?- reemplazar(1, a, 2, 2, [1, 1, 2, 2, 3, 3, 2, 2, 1, 1], L).
% L = [1, a, 2, 2, 3, 3, 2, 2, a, 1]

?- reemplazar(1, a, 2, -1, [1, 1, 2, 2, 3, 3, 2, 2, 1, 1], L).
% L = [1, a, 2, 2, 3, 3, 2, 2, a, a]

?- reemplazar(1, a, 5, -1, [1, 1, 2, 2, 3, 3, 2, 2, 1, 1], L).
% L = [1, 1, 2, 2, 3, 3, 2, 2, 1, 1]

?- reemplazar(4, a, 1, -1, [1, 1, 2, 2, 3, 3, 2, 2, 1, 1], L).
% L = [1, 1, 2, 2, 3, 3, 2, 2, 1, 1]
```

Instrucciones:

1. Defina el predicado reemplazar/6.
2. Valide los parámetros para asegurar que cumplen con las condiciones especificadas.
3. Implemente la lógica para reemplazar elementos en la lista según las reglas dadas.

Ejercicio 8: Encontrar la Salida en un Plano de una Casa

Cree un programa que encuentre la salida dado el plano de una casa. Para esto:

a) Debe resolver cómo modelar el plano.

b) Debe encontrar la forma de avanzar en el camino hacia la salida.

c) Debe verificar que no se formen bucles en el camino.

Ejemplo

```
?- salir(a, Camino).
% Camino = [a, b, c, g, Salida];
% Camino = [a, b, e, f, g, Salida];
% Camino = [a, d, f, g, Salida];
% Camino = [a, d, f, e, b, c, g, Salida].
```

Instrucciones:

1. Modele el plano de la casa utilizando una estructura adecuada en Prolog.
2. Implemente el predicado salir/2 para encontrar un camino hacia la salida.
3. Asegúrese de que el camino encontrado no forme bucles.