

Trabajo Práctico 2. Programación Lógica

Puntuación

Puntaje Total: 100 puntos

Aprobación: 60 puntos

Fecha de entrega: 09/11/2025 – 23:59 Hs.

Condiciones de entrega

1. Para resolver este trabajo práctico **la conformación del grupo debe ser la misma del Trabajo Práctico 1.**
2. **Entrega:** Se realizará por medio del Campus Virtual de la UTN, en la tarea correspondiente al TP 2. La extensión del archivo será .zip o .rar, de acuerdo al programa de compresión usado. El nombre del archivo se consigue concatenando un prefijo del número del TP con los apellidos de los integrantes por orden alfabético y separados por guiones (por ejemplo: para Nuñez, Acevedo, González y Peralta, el nombre será: TP2-Acevedo-González-Nuñez-Peralta.zip). El nombre de archivo no debe contener espacios. Dentro del archivo de entrega, deben constar los siguientes:
 - **Fuentes SWI-Prolog:** Se debe entregar un archivo denominado TP2.pl con el código de todos los predicados solicitados, adecuadamente comentado.
 - **Método adicional Smalltalk solicitado:** Incluir un archivo conteniendo el método específico que se solicita en el enunciado. Generar el archivo utilizando la opción #fileout del menú desplegable del método, con el nombre: CentroTrasplantes-guardaBaseDeDatosProlog.st.
 - Los **Casos de Prueba** se entregarán en un archivo de texto, no deben ser capturas de pantalla. Deberán cubrir diferentes resultados que puedan obtenerse según los predicados solicitados. Se enfatiza que se adjunten casos de prueba que sean claros, válidos y suficientes para poder probar el trabajo. Entregar este archivo con el nombre casos-de-prueba.txt.
 - **Utilización de modelos de lenguaje (LLMs):** en caso de utilizar estas herramientas se entregará un archivo de texto (.txt) o markdown (.md o .markdown) con un listado describiendo brevemente los predicados que se analizaron o resolvieron mediante conversaciones con modelos de IA generativa (ChatGTP, Gemini, etc.). Incluir al final de este archivo una sección con conclusiones. Archivo: consultas-IA.markdown.
 - **Archivo de integrantes** (integrantes.txt) con una línea para cada integrante en la cual figure el nombre del alumno/a y su dirección de email.
3. **Penalización por entrega fuera de término:** Si el trabajo práctico se entrega después de la fecha indicada, y hasta una semana tarde, tendrá una quita de 15 puntos. No serán recibidos trabajos luego de una semana de la fecha de entrega. Los trabajos prácticos para los cuales se solicite rehacer/corregir fuera de la fecha de entrega tienen una quita de 30 puntos.

Aclaración importante:

Para la realización de este trabajo práctico **pueden utilizarse únicamente los predicados predefinidos de Prolog vistos en clase**, además del predicado **string_concat/3** explícitamente indicado. En otras palabras: no utilizar **;/2**, **->/2**, **findall/3**, etc.

Introducción – Cadenas de Trasplantes de Riñón

La **insuficiencia renal crónica** consiste en el deterioro progresivo e irreversible de la función renal. Para los pacientes que sufren esta patología, los trasplantes de riñón salvan miles de vidas cada año. La gran dificultad de estos trasplantes es conseguir un donante compatible con el paciente. Si el donante es incompatible, el cuerpo del paciente rechazará el riñón donado y la situación resultante será peor que la del principio. El problema de evaluar la compatibilidad entre donantes y receptores fue abordado en el Trabajo Práctico 1.

Como alternativa para remediar su situación, un paciente puede proponer un familiar (o conocido) que está dispuesto a realizar la donación. Si el paciente y su donante son compatibles, se planifica la cirugía y el paciente recibe el riñón de su donante sin inconvenientes. Sin embargo, también es posible que el donante sea incompatible con su propio pariente o conocido, pero tenga una compatibilidad elevada con otro paciente que busca un riñón.

A fin de decidir qué trasplantes serán realizados, cuando un paciente y su donante registrado son incompatibles, es posible encontrar otras soluciones para abordar el problema. Supongamos dos pacientes y sus donantes registrados: se puede dar el caso que los pacientes intercambien donantes, y que así cada donante dona un riñón, y cada paciente recibe un riñón. En este caso se trata de una cadena de trasplantes de longitud 2. Siguiendo la misma lógica, pueden realizarse también cadenas de longitud mayor, como 3, 4, etc. La condición que debe darse siempre es que, cuando el donante de un paciente entrega su riñón, su paciente asociado también recibe un riñón.

Para que esta condición se cumpla, obviamente, será necesario realizar los trasplantes de manera simultánea, y así evitar que se arrepienta el donante de un paciente que ya recibió un riñón y se corte la cadena. Los trasplantes simultáneos para una gran cantidad de pacientes tienen el inconveniente de requerir una gran infraestructura de quirófanos y profesionales disponibles simultáneamente. Es así que cadenas más largas tienen la ventaja de involucrar a más pacientes, y por lo tanto resolver el problema de más personas con insuficiencia renal, pero también tienen la desventaja de ser mucho más dificultosas de llevar a la práctica.

Dados un grupo de pacientes y sus donantes registrados, el objetivo del Trabajo Práctico 2 es **implementar un conjunto de predicados Prolog que permita hallar cadenas de trasplantes adecuadas para los pacientes y donantes de una base de datos disponible**, ampliando las capacidades del sistema de gestión del centro de trasplantes desarrollado en el Trabajo Práctico 1. Resolver este problema es un ejemplo de aplicación de Tecnología Informática en Salud.

Exportar información de pacientes, donantes y compatibilidades

El **Centro de Trasplantes** deberá contar con una **Base de Datos Prolog** con la información de los pacientes que sufren insuficiencia renal crónica, los donantes registrados para dichos pacientes, y las compatibilidades evaluadas clínicamente entre ellos. La información de los pacientes se representará con una lista de hechos tales como:

```
paciente(dni(23100200), nombre("Juan", "Martínez"), tiempo_espera(18)).  
paciente(dni(37123456), nombre("Ariel", "Lencina"), tiempo_espera(15)).  
paciente(dni(27561894), nombre("Ana", "Ramos"), tiempo_espera(8)).  
paciente(dni(25450295), nombre("Fernando", "Pérez"), tiempo_espera(20)).
```

Donde *Juan Martínez* tiene DNI 23.100.200 y se encuentra en lista de espera para un trasplante hace 18 meses, siendo similar la interpretación para los demás hechos. Por su parte, los donantes registrados para los pacientes se representan con hechos tales como:

```
donante(dni(25200100), nombre("Luis", "Martínez"), dni_paciente(23100200)).  
donante(dni(33292500), nombre("Pedro", "Sánchez"), dni_paciente(37123456)).  
donante(dni(26854963), nombre("María", "Ramos"), dni_paciente(27561894)).  
donante(dni(24700600), nombre("Andrea", "Vera"), dni_paciente(25450295)).
```

En este caso, el primer hecho representa que *Luis Martínez* está registrado como donante de *Juan Martínez*. De modo similar se interpretan los demás hechos.

Para determinar si un trasplante puede realizarse o no es necesario considerar la compatibilidad entre el donante y el paciente, problema abordado previamente en el Trabajo Práctico 1. Los resultados de compatibilidad se representarán en *Prolog* del siguiente modo:

```
compatibilidad(25200100, 37123456, 80.0).  
compatibilidad(25200100, 27561894, 55.0).  
compatibilidad(25200100, 25450295, 75.0).  
compatibilidad(33292500, 23100200, 90.0).  
compatibilidad(33292500, 25450295, 58.0).  
compatibilidad(26854963, 23100200, 60.0).  
compatibilidad(26854963, 37123456, 72.0).  
compatibilidad(26854963, 27561894, 20.0).  
compatibilidad(26854963, 25450295, 65.0).  
compatibilidad(24700600, 37123456, 70.0).  
compatibilidad(24700600, 27561894, 85.0).
```

Es decir, que *Luis Martínez* es compatible con *Ariel Lencina* en un 80%, con *Ana Ramos* en un 55%, etc.

Los hechos anteriores incluyen, para cada donante, los resultados de compatibilidad con varios pacientes. Cuando no se dispone de información o la compatibilidad es menor al 50%, se considera que el donante y el paciente son incompatibles.

Se solicita lo siguiente:

- 1) **Implementar en Pharo Smalltalk**, complementando lo desarrollado en el Trabajo Práctico 1, el método siguiente:

Clase CentroTrasplantes – Método de instancia	
guardaBaseDeDatosProlog	<p>Genera un archivo con código Prolog denominado <i>NombreDeLCentro.pl</i> incluyendo todos los hechos descriptos previamente para pacientes, donantes y compatibilidades.</p> <p>En forma similar al método <code>guardaRegistros:</code> del TP1, en este caso la información se exportará en un único archivo de texto con extensión <code>.pl</code>, el cual incluye el código Prolog que describe los hechos de la base de datos.</p>

- 2) Proponer al menos **3 archivos de Base de Datos Prolog** para evaluar distintos escenarios de compatibilidades, que generen alternativas para las cadenas de trasplantes halladas, **incluyendo cadenas de longitud 8 en alguno de los ejemplos**.

Criterios para decidir trasplantes

Dados los pacientes y donantes compatibles, se desea incorporar un criterio adicional para determinar la prioridad que se le dará a cada trasplante posible. Esta prioridad permitirá cuantificar la urgencia del trasplante. La fórmula a utilizar será:

$$\text{Prioridad} = \text{Compatibilidad} + \Delta * \text{TiempoEsperaEnAños}$$

Donde Δ es una constante, la cual es multiplicada por *TiempoEsperaEnAños*, que corresponde al tiempo de espera del paciente convertido de meses a años, sin redondear. Por ejemplo: para el caso de *Pedro Sánchez* y *Juan Martínez*, tomando $\Delta = 2.0$ tenemos:

$$\text{Prioridad} = 90.0 + 2.0 * (18 / 12) = 93.0$$

Notar que el trasplante de riñón de *Pedro* a *Juan* tendrá una prioridad mayor a la compatibilidad entre ambos, donde el factor $\Delta = 2.0$ representa un incremento del 10% cada 5 años en lista de espera. El objetivo buscado con esta fórmula es favorecer a los pacientes que llevan más tiempo esperando un trasplante. Para definir la constante Δ se incluye el siguiente hecho en el programa:

`delta(2.0).`

Además, al momento de decidir qué pacientes deben recibir un riñón, se puede considerar obligatorio realizar un trasplante a aquellos que llevan más tiempo en la lista de espera. En este caso, se considerará un umbral U , de modo que todos los pacientes con tiempo de espera mayor o igual a U deban ser trasplantados.

Representación de cadenas de trasplantes

Como se mencionó previamente, al permitir donaciones a pacientes diferentes al familiar registrado, es posible obtener cadenas de trasplantes de distinta longitud. Para armar una cadena de trasplantes, una condición fundamental es la siguiente: si un paciente tiene un donante registrado, y ese donante dona su riñón a un tercero, entonces el paciente mencionado debe recibir también un riñón dentro de la misma cadena de asignación.

Para los datos planteados más arriba, podemos considerar como ejemplo la siguiente cadena: *Luis* dona un riñón a *Ana*, *María* dona un riñón a *Ariel*, y *Pedro* dona un riñón a *Juan*. Si utilizamos la lista **[Donante, Paciente]** para representar cada trasplante, los trasplantes mencionados se representarán con la lista de listas siguiente:

```
[[donante(25200100, "Luis Martínez"), paciente(27561894, "Ana Ramos")],
 [donante(26854963, "María Ramos"), paciente(37123456, "Ariel Lencina")],
 [donante(33292500, "Pedro Sánchez"), paciente(23100200, "Juan Martínez")]]
```

donde cada paciente que recibe un riñón tiene un donante registrado que entrega un riñón. Nótese que **Donante** y **Paciente** son, respectivamente, funciones **donante/2** y **paciente/2** cuyos argumentos son los DNI y "Nombre Apellido" de las personas involucradas.¹

En general, si D_i es el donante registrado para el paciente P_i , una cadena de N trasplantes tendrá la forma:

$$[[D_1, P_2], [D_2, P_3], \dots, [D_{N-1}, P_N], [D_N, P_1]]$$

Predicados solicitados

A fin de obtener y comparar las diferentes alternativas de trasplantes para un grupo de donantes y pacientes, se solicita definir los siguientes predicados en **Prolog**:

registrado(**Donante**, **Paciente**)

Este predicado acierta cuando **Donante** es el donante registrado de **Paciente**.

compatible(**Donante**, **Paciente**)

Este predicado es exitoso cuando la compatibilidad entre **Donante** y **Paciente** es mayor o igual al 50%, lo cual significa que puede realizarse el trasplante. El **Donante** puede estar registrado para cualquier paciente.

prioridad(**Donante**, **Paciente**, **P**)

Este predicado permite obtener la prioridad **P** que tendrá la donación de un riñón de **Donante** a **Paciente**, de acuerdo a la fórmula indicada previamente y utilizando el

¹ Para unir cadenas de caracteres utilice el predicado **string_concat/3** disponible en Prolog.

valor Δ definido en el programa. El predicado falla si **Donante** y **Paciente** no son compatibles.

lista_pacientes(Umbra1, ListaPacientes)

Permite obtener la lista de pacientes en la base de datos que tienen al menos **Umbra1** meses de tiempo de espera.

trasplantes(ListaPacientes, ListaTrasplantes)

Permite hallar una cadena de asignación de trasplantes, representada por **ListaTrasplantes**, para los pacientes en **ListaPacientes**. Debe evitar la obtención de soluciones redundantes.

prioridad_minima(ListaTrasplantes, Min)

Permite obtener la prioridad **Min**, correspondiente al trasplante con menor prioridad, dentro de la cadena de trasplantes **ListaTrasplantes**.

cadena_trasplantes_optima(ListaPacientes, ListaTrasplantes)

Obtiene una cadena de asignación de trasplantes *óptima*, representada por **ListaTrasplantes**, para los pacientes en **ListaPacientes**. Una cadena *óptima* será aquella que, considerando todas las cadenas de trasplantes posibles para los pacientes involucrados, tenga la mayor prioridad mínima posible.

cadena_trasplantes_completa(Umbra1, ListaTrasplantes)

Obtiene una cadena de trasplantes *óptima* que incluye a todos los pacientes con al menos **Umbra1** meses en lista de espera.

En los predicados anteriores, **ListaTrasplantes** es una lista que contiene sublistas de 2 elementos, que a su vez representan asignaciones compatibles entre donante y paciente, como se indicó previamente.

Al utilizar una lista para representar una cadena de trasplantes surge el inconveniente que cualquier rotación de la lista (mover elementos del principio al final, o viceversa) representa la misma cadena de trasplantes. Para evitar la obtención de soluciones redundantes, empiece la cadena con el donante registrado del primer paciente en la lista de pacientes de la consulta.

Ejemplos

A continuación se presentan algunos ejemplos para la base de datos propuesta:

?- **registrado**(D, P).

```
D = donante(25200100, "Luis Martínez"),
P = paciente(23100200, "Juan Martínez") ;
D = donante(33292500, "Pedro Sánchez"),
P = paciente(37123456, "Ariel Lencina") ;
```

```
D = donante(26854963, "María Ramos"),
P = paciente(27561894, "Ana Ramos") ;
D = donante(24700600, "Andrea Vera"),
P = paciente(25450295, "Fernando Pérez").
```

```
?- compatible(D, paciente(23100200, "Juan Martínez")).
D = donante(33292500, "Pedro Sánchez") ;
D = donante(26854963, "María Ramos") ;
false.
```

```
?- compatible(donante(25200100, "Luis Martínez"), P).
P = paciente(37123456, "Ariel Lencina") ;
P = paciente(27561894, "Ana Ramos") ;
P = paciente(25450295, "Fernando Pérez") ;
false.
```

```
?- prioridad(donante(33292500, "Pedro Sánchez"), paciente(23100200, "Juan
Martínez"), X).
X = 93.0.
```

```
?- prioridad(donante(33292500, NA), P, X).
NA = "Pedro Sánchez",
P = paciente(23100200, "Juan Martínez"),
X = 93.0 ;
NA = "Pedro Sánchez",
P = paciente(25450295, "Fernando Pérez"),
X = 61.333333333333336.
```

```
?- lista_pacientes(10, LP).
LP = [paciente(23100200, "Juan Martínez"), paciente(37123456, "Ariel
Lencina"), paciente(25450295, "Fernando Pérez")].
```

```
?- trasplantes([paciente(23100200, "Juan Martínez"), paciente(37123456,
"Ariel Lencina"), paciente(27561894, "Ana Ramos")], LT).
LT = [[donante(25200100, "Luis Martínez"), paciente(27561894, "Ana
Ramos")], [donante(26854963, "María Ramos"), paciente(37123456, "Ariel
Lencina")], [donante(33292500, "Pedro Sánchez"), paciente(23100200, "Juan
Martínez")]] ;
false.
```

```
?- trasplantes([paciente(37123456, "Ariel Lencina"), paciente(27561894,
"Ana Ramos")], LT).
false.
```

```
?- trasplantes([paciente(23100200, "Juan Martínez"), paciente(37123456,
"Ariel Lencina"), paciente(27561894, "Ana Ramos"), paciente(25450295,
"Fernando Pérez")], LT).
```

```
LT = [[donante(25200100, "Luis Martínez"), paciente(37123456, "Ariel
Lencina")], [donante(33292500, "Pedro Sánchez"), paciente(25450295,
"Fernando Pérez")], [donante(24700600, "Andrea Vera"), paciente(27561894,
"Ana Ramos")], [donante(26854963, "María Ramos"), paciente(23100200, "Juan
Martínez")]] ;
```

```
LT = [[donante(25200100, "Luis Martínez"), paciente(27561894, "Ana
Ramos")], [donante(26854963, "María Ramos"), paciente(25450295, "Fernando
Pérez")], [donante(24700600, "Andrea Vera"), paciente(37123456, "Ariel
Lencina")], [donante(33292500, "Pedro Sánchez"), paciente(23100200, "Juan
Martínez")]] ;
```

```
LT = [[donante(25200100, "Luis Martínez"), paciente(25450295, "Fernando
Pérez")], [donante(24700600, "Andrea Vera"), paciente(27561894, "Ana
Ramos")], [donante(26854963, "María Ramos"), paciente(37123456, "Ariel
Lencina")], [donante(33292500, "Pedro Sánchez"), paciente(23100200, "Juan
Martínez")]] ;
```

```
false.
```

```
?- prioridad_minima([[donante(25200100, "Luis Martínez"),
paciente(27561894, "Ana Ramos")], [donante(26854963, "María Ramos"),
paciente(37123456, "Ariel Lencina")], [donante(33292500, "Pedro Sánchez"),
paciente(23100200, "Juan Martínez")]], Min).
```

```
Min = 56.333333333333336 ;
```

```
false.
```

```
?- cadena_transplantes_optima([paciente(23100200, "Juan Martínez"),
paciente(37123456, "Ariel Lencina"), paciente(27561894, "Ana Ramos"),
paciente(25450295, "Fernando Pérez")], LT).
```

```
LT = [[donante(25200100, "Luis Martínez"), paciente(25450295, "Fernando
Pérez")], [donante(24700600, "Andrea Vera"), paciente(27561894, "Ana
Ramos")], [donante(26854963, "María Ramos"), paciente(37123456, "Ariel
Lencina")], [donante(33292500, "Pedro Sánchez"), paciente(23100200, "Juan
Martínez")]] ;
```

```
false.
```

```
?- cadena_trasplantes_completa(10, LT).
```

```
LT = [[donante(25200100, "Luis Martínez"), paciente(25450295, "Fernando
Pérez")], [donante(24700600, "Andrea Vera"), paciente(37123456, "Ariel
Lencina")], [donante(33292500, "Pedro Sánchez"), paciente(23100200, "Juan
Martínez")]] ;
```

```
false.
```