PEP 2 Paradigmas de Programación (Mayo 2022)

NOMBRE: **PROFESOR:** **RUT:**

Instrucciones: En **AMBAS PREGUNTAS** **(1)** hacer un uso adecuado del paradigma lógico (según corresponde,**(2)** use pseudo-prolog **(3)** documentar; **(4)** responder preguntas en hojas separadas indicando en cada una (preguntas y respuestas) su nombre, rut y profesor; **(5)** entregar todas las hojas (preguntas y respuestas) al final; **(6)** no es necesario comprobar tipos; y **(7)** solo puede usar los elementos más primitivos del lenguaje (operadores aritméticos, lógicos, sintaxis del lenguaje). No puede usar predicados especializados de SWI Prolog (ej. para manipular listas). Si los requiere, debe implementarlos.

Pregunta 1 (30 pts):

Considere la especificación e implementación parcial del TDA document, a partir de los siguientes predicados, el cual forma parte de un sistema de ofimática como Word o GoogleDocs.

a. (1.5 pts) ;consulta por la primera palabra de un documento.
%dominios: document X string
%firstWord(Document,FirstWord)

b. (1.5 pts) ;consulta por las siguientes palabras después de la primera.
%dominios: document X Document
%nextWords(Document,NextWords)

c. (5 pts) ;consulta las posiciones donde se ubica una palabra en el documento. La variable de salida corresponde a una lista con todas las posiciones donde figura la palabra.
%dominios: document X string X int list
%find(Document,Word,PositionList)

d. (5 pts) ;encuentra el documento resultante luego de reemplazar todas las ocurrencias de la palabra especificada en word, por newWord.
%dominios: document X string X string X document
%replaceAll(DocumentIn,Word,NewWord,DocumentOut)

e. (7 pts) ;consulta por la cantidad de caracteres en el documento sin considerar los espacios (se asume que entre palabras existen espacios simples, es decir: " "). Puede usar string_length(String,Count)
%dominios: document X int
%charactersCountWithoutSpace(Document,Count)

f. (10 pts) ;busca frases (ej: "esta es una frase") en un documento. La variable de salida corresponde a una lista con las posiciones donde comienzan las frases que coinciden.
%dominios: document X string X int list
%findPhrase(Document,Phrase,PositionList)

Respecto de la implementación parcial del TDA documento, se conoce su representación.

document = null | string X document

y el constructor base buildDoc(Texto,Document), donde Texto=string y Document es la lista de strings resultantes. ej: buildDoc("hola mundo",Doc) => Doc = ["hola"," ","mundo"].
A partir de los antecedentes provistos, implemente cada una de las operaciones del TDA señaladas en los ítems (a) al (f).

PEP 2 Paradigmas de Programación (Mayo 2022)

NOMBRE: **PROFESOR:** **RUT:**

Pregunta 2 (30 pts):

Se planea construir un sistema para un servicio de transporte urbano (como Uber o Lyft) que permita vincular usuarios clientes con usuarios conductores a través de viajes. En particular, un **usuario cliente** registrado con un correo electrónico puede solicitar un **viaje** ingresando una dirección origen y una dirección destino. Respecto del **viaje**, este consta de un identificador, el correo electrónico del conductor que acepta el viaje, el correo del cliente que solicita el viaje, las direcciones de origen y destino, distancia total a recorrer representada en kilómetros (km), tarifa por km recorrido, estado del viaje (ej: asignado, rechazado, terminado), duración del viaje en minutos y finalmente el precio a pagar CLP.

En relación al **usuario conductor,** éste está registrado con un correo electrónico y cuenta con un registro histórico de todos los viajes que ha realizado y aquellos que se le han asignado (no necesariamente concretado). Estos últimos se van actualizando conforme el conductor acepta o rechaza los viajes que se le asignen.

A partir de estos antecedentes:

a) (**3 pts**) Especificar el o los TDA(s) para abordar la construcción del sistema. En la especificación de cada TDA (listado de operaciones), sólo considere lo absolutamente necesario para cubrir los requerimientos de esta pregunta.
b) (**6 pts**) Implementar el o los TDA(s) identificados. Procure señalar la representación escogida para todos los TDAs. Luego, implemente los predicados expresados en (a) (Solo aquellos absolutamente necesarios para responder a los siguientes ítems).
c) (**4 pts**) Implementar un predicado que permita obtener el total de km recorridos de un conductor considerando solo los viajes realizados.
d) (**2 pts**) Implementar un predicado que permita obtener el precio total de un viaje. Esto calculado a partir de los kilómetros recorridos y la tarifa.
e) (**15 pts**) Implementar un predicado que permita a un cliente determinar el o los viajes que reúnan ciertas condiciones (ej: el más largo que X, el más costoso que X, el más barato que X, etc.) las que quedan expresadas a través de un predicado que se pasa como variable al momento de la consulta. Para estos efectos, considere que para usar un predicado pasado como variable en una cláusula, debe usar el predicado *call* que opera de la siguiente manera.

```
aplicarPredicado(Pred,V):-call(Pred,V).
aplicarPredicado(string,5).   %equivalente a consultar string(5).
                              %para determinar si 5 es un string
```

Para autoevaluación recuerde evaluar cada item de acuerdo a la siguiente escala de apreciación:

Ptje	Descripción
0	No responde pregunta. Respuesta no corresponde con la pregunta.
.0.25	Respuesta incompleta con errores mayores pero que revela cierta orientación hacia la solución del problema.
0.5	Respuesta incompleta con errores mayores. Demuestra entendimiento del problema, sin embargo la aplicación del paradigma y el lenguaje es imprecisa o incorrecta. Ej: omite casos base, descomposición recursiva, no aplica lo solicitado para la pregunta, etc.
0.75	Respuesta completa con errores menores. Si bien la respuesta es completa, presenta algunas imprecisiones. Demuestra un correcto uso del paradigma y lenguaje.
1	Respuesta completa sin errores

Luego la nota se calcula de la siguiente forma:

$$Nota = 1 + (\sum_{i} Pi * PtjeTotalITEMi)/10 \quad ; Pi\ es\ el\ ptje\ asignado\ en\ la\ evaluación\ para\ el\ item\ i$$

Pauta Pregunta 1

a. (1.5 pts) ;consulta por la primera palabra de un documento.
%dominios: document X string
firstWord([FirstWord|_],FirstWord).

b. (1.5 pts) ;consulta por las siguientes palabras después de la primera.
%dominios: document X Document
nextWords([_|NextWords],NextWords).

c. (5 pts) ;consulta las posiciones donde se ubica una palabra en el documento. La variable de salida corresponde a una lista con todas las posiciones donde figura la palabra.
%dominios: document X string X int list
find(Document,Word,PositionList):-findAux(Document,Word,0,PositionList).

findAux([],_,_,[]).
findAux([W|Ws],W,P,[P|Ps]):-NextP is P+1, findAux(Ws,W,NextP,Ps).
findAux([_|Ws],W,P,Ps):-NextP is P+1, findAux(Ws,W,NextP,Ps).

d. (5 pts) ;encuentra el documento resultante luego de reemplazar todas las ocurrencias de la palabra especificada en word, por newWord.
%dominios: document X string X string X document
replaceAll([],_,_,[]).
replaceAll([W|Ws],W,NewW,[NewW|NWs]):-replaceAll(Ws,W,NewW,NWs).
replaceAll([W2|Ws],W,NewW,[W2|NWs]):-W\=W2,replaceAll(Ws,W,NewW,NWs).

e. (7 pts) ;consulta por la cantidad de caracteres en el documento sin considerar los espacios (se asume que entre palabras existen espacios simples, es decir: " "). Puede usar string_length(String,Count)

```
%dominios: document X int
;una forma de hacerlo sin maplist
charactersCountWithoutSpace(Document,Count):-replaceAll(Document," ","",NewDoc),
                                             countAux(NewDoc,Count).

countAux([],0).
countAux([W|Ws],Count):-string_length(W,C),countAux(Ws,C2),Count is C2 + C.
```

f. (10 pts) ;busca frases (ej: "esta es una frase") en un documento. La variable de salida corresponde a una lista con las posiciones donde comienzan las frases que coinciden.

```
%dominios: document X string X int list
findPhrase(Doc,Phrase,PosList):-
                    buildDoc(Phrase,PhraseList),findPhraseAux(Doc,PhraseList,0,PosList).

findPhraseAux([],_,_,[]):-!.
findPhraseAux([W|Ws],Phrase,Pos,[Pos|PL]):-isphrase([W|Ws],Phrase),NewP is Pos+1,
                                findPhraseAux(Ws,Phrase,NewP,PL).
findPhraseAux([_|Ws],Phrase,Pos,PosList):-NewP is Pos+1,
                                findPhraseAux(Ws,Phrase,NewP,PosList).

isphrase(_,[]).
isphrase([W|Ws],[W|OWs]):-isphrase(Ws,OWs).
```

Pauta Pregunta 2

a) (**3 pts**) Especificar el o los TDA(s) para abordar la construcción del sistema. En la especificación de cada TDA (listado de operaciones), sólo considere lo absolutamente necesario para cubrir los requerimientos de esta pregunta.
b) (**2 pts**) Implementar un predicado que permita obtener el total de km recorridos de un conductor considerando solo los viajes realizados.
c)
d) (**15 pts**) Implementar un predicado que permita a un cliente determinar el o los viajes que reúnan ciertas condiciones (ej: el más largo que X, el más costoso que X, el más barato que X, etc.) las que quedan expresadas a través de un predicado que se pasa como variable al momento de la consulta. Para estos efectos, considere que para usar un predicado pasado como variable en una cláusula, debe usar el predicado *call* que opera de la siguiente manera.

```
aplicarPredicado(Pred,V):-call(Pred,V).
aplicarPredicado(string,5).    %equivalente a consultar string(5).
                               %para determinar si 5 es un string
```

e)

TDA sistema
TDA cliente
- Otras: filterViajesCliente

TDA clientes
TDA conductor
- selectores: getViajes(eConductor,Viajes)~~:viajes~~
- otras: getTotalKmRecorridos

TDA conductores
TDA viaje
- Selectores: distanciaRecorrida
-
  tarifa
- Otras: viajeTerminado,
- precioTotalViaje

TDA viajes

f) (**6 pts**) Implementar el o los TDA(s) identificados. Procure señalar la representación escogida para todos los TDAs. Luego, implemente los predicados expresados en (a) (Solo aquellos absolutamente necesarios para responder a los siguientes ítems).

TDA sistema. Representación: Clientes X Conductores
TDA cliente: Representación Correo (String) X Viajes
- predicados implementados para selectores y otras operaciones se presentan junto al desarrollo de los siguientes items.

TDA conductor: Representación: Correo (String) X Viajes

- predicados implementados para selectores y otras operaciones se presentan junto al desarrollo de los siguientes items.

TDA viaje: Representación: id (Integer) X Cliente X Conductor X Estado~~Aceptación~~ (~~boolean~~symbol {terminado,iniciado, otros …}) X Origen (String) X Destino (String) X DistanciaTotal (Integer) X Tarifa (Integer) X Duración (Integer) X Precio (Integer)

- predicados implementados para selectores y otras operaciones se presentan junto al desarrollo de los siguientes items~~}~~.

TDA viajes: null | viaje X viajes

TDA clientes: null | cliente X clientes
TDA conductores: null | conductor X conductores|

g) (**4 pts**) Implementar un predicado que permita obtener el total de km recorridos de un conductor considerando solo los viajes realizados.

```
getTotalKmRecorridos([Email,Viajes],Kms):-getTotalKmRecorridosAux(Viajes,Kms).
getTotalKmRecorridosAux([],0).
getTotalKmRecorridosAux([V|Vs],Kms):-viajeTerminado(V),
                                     distanciaRecorrida(V,D),
                                     getTotalKmRecorridosAux(Vs,Kms2),
                                     Kms is Kms2 + D.
getTotalKmRecorridosAux([V|Vs],Kms):-not(viajeTerminado(V)),
                                     getTotalKmRecorridosAux(Vs,Kms).

distanciaRecorrida([_,_,_,_,_,_,D|_],D).
viajeTerminado([_,_,_,terminado|_]).
```

h) (**2 pts**) Implementar un predicado que permita obtener el precio total de un viaje. Esto calculado a partir de los kilómetros recorridos y la tarifa.

```
precioTotalViaje(V,Total):-distanciaRecorrida(V,Km),tarifa(V,T),Total is Km*T.

tarifa([_,_,_,_,_,_,_,T|_],T).
```

i)

```
precioTotalViaje(V,Total):-distanciaRecorrida(V,Km),tarifa(V,T),Total is Km*T.

tarifa([_,_,_,_,_,_,_,T|_],T).
```

j) (**15 pts**) Implementar un predicado que permita a un cliente determinar el o los viajes que reúnan ciertas condiciones (ej: el más largo que X, el más costoso que X, el más barato que X, etc.) las que quedan expresadas a través de un predicado que se pasa como variable al momento de la consulta. Para estos efectos, considere que para usar un predicado pasado como variable en una cláusula, debe usar el predicado *call* que opera de la siguiente manera.

```
aplicarPredicado(Pred,V):-call(Pred,V).
aplicarPredicado(string,5).    %equivalente a consultar string(5).
```

%para determinar si 5 es un string

```
filterViajesCliente(Crit,C,Vs):-viajesCliente(C,VC),filterViajesClienteAux(Crit,VC,Vs).

filterViajesClienteAux(_,[],[]).
filterViajesClienteAux(Crit,[V|Vs],[V|NVs]):-call(Crit,V),
                                              filterViajesClienteAux(Crit,Vs,NVs).
filterViajesClienteAux(Crit,[V|Vs],NVs):-
                                          filterViajesClienteAux(Crit,Vs,NVs).
```

%Evaluar documentación según escala de apreciación en base a nivel de completitud.
%:1 completo, .75: faltan algunos elementos, .5: incompleta, faltan predicados, variables, y/o metas, .25: precaria, solo algunos indicios de documentación, 0: no documenta.
% El puntaje obtenido en este apartado se opera de la siguiente manera
% PtjeDescuentoDocumentación = (1-PtjeAlcanzado)*10. (Se descuenta hasta 1 pto de la prueba por no documentar)

%Documentación

%Dominios
%Conductor: Conductor
%Cliente: Cliente
%Viajes: Viajes
%Viaje: Viaje
%Kilometros: Número+ U {0}
%Distancia: Número+ U {0}
%Tarifa: Número+ U {0}
%Precio: Número+ U {0}

%Predicados
%getTotalKmRecorridos(Conductor,Kilometros)
%distanciaRecorrida(Viaje,Distancia)
%viajeTerminado(Viaje)
%precioTotalViaje(Viaje,Precio)
%tarifa(Viaje,Tarifa)
%filterViajesCliente(Predicado,Cliente,Viajes)
%filterViajesClienteAux(Predicado,Viajes,Viajes)

%Metas
%principal
%filterViajesCliente,precioTotalViaje,getTotalKmRecorridos

%secundarias

%tarifa, filterViajesClienteAux, viajeTerminado, distanciaRecorrida

%clausulas
%desarrolladas en los apartados anteriores