PEP 2 Paradigmas de Programación (2021-1)

NOMBRE:_____ RUT: _____ Profesor: _____

Instrucciones

- Tiempo estimado de respuesta: 90 minutos
- Tiempo disponible para entregar respuestas desde que tuvo acceso al enunciado: 3 horas hasta las 23:59 del 10 de junio de 2021.
- Lugar de entrega: Campus Virtual.
- Escriba sus respuestas en hojas separadas
- Puede escribir sus respuestas directamente en un documento electrónico o bien puede hacerlo en papel. Escoja el que más le acomode.
- Recomendación: Para asegurar acceso continuo a este enunciado en caso de cortes de suministro eléctrico o en el acceso a Internet, procure descargar este enunciado y si puede, imprimalo.
- Identifique cada hoja de respuestas con su nombre y rut, lo que sirve de complemento a la declaración que ha aceptado en Campus Virtual.
- Identifique cada hoja de respuesta con el profesor de su sección.
- Al terminar la evaluación,
    - Si desarrolló sus respuestas con lápiz y papel, saqué una fotografía o escanee cada una de sus hojas de respuestas. Asegúrese que las imágenes de sus capturas son legibles (abra el archivo y verifique). Luego, agrupe todas las imágenes en un PDF (en Windows basta con seleccionar las imágenes desde el explorador de archivos -> clic derecho -> Imprimir -> Escoger Impresora PDF de Microsoft u otra que disponga) o bien en un archivo comprimido *.zip* o *.rar*.
    - Si desarrolló sus respuestas directamente en formato digital (ej: documento Word, Google Docs, Latex, etc.) guardar el documento como PDF. Se recomienda este formato para evitar problemas en la visualización de estilos.
- Finalmente, subir el archivo al espacio creado para tales efectos en Campus Virtual. Una vez subido el archivo, descárguelo y asegúrese que el archivo se puede abrir (para evitar situaciones de archivos dañados).
- Excepción: En la eventualidad que no pueda subir su trabajo a Campus Virtual en el plazo disponible, enviarlo inmediatamente vía correo a su profesor.

Sobre la evaluación. Todo ítem se evalúa en una escala de 3 puntos: 0 (sin respuesta o respuesta no aborda el problema), 0.5, 1 (respuesta completa sin errores), según grado de cumplimiento.

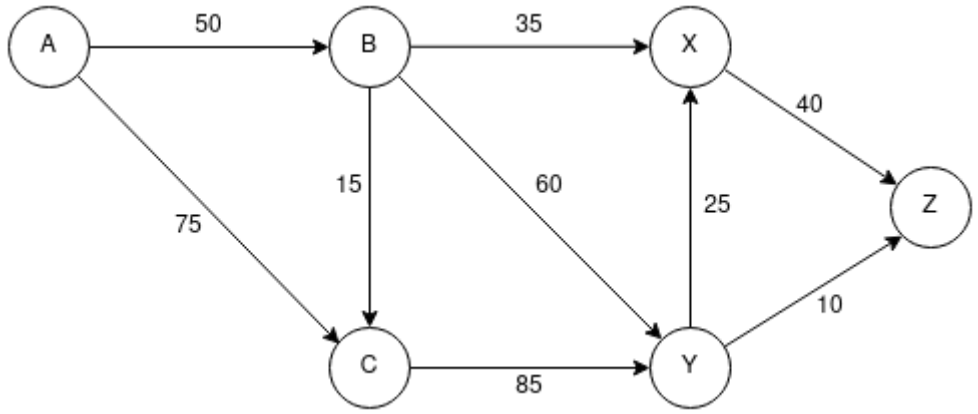
NOTA 1: Procure hacer un adecuado uso adecuado de la programación declarativa y particularmente de la programación lógica. La implementación innecesaria y/o redundante de predicados que desaprovechan el paradigma lógico tendrán una penalización de hasta 2 puntos por cada ítem afectado.

Nota 2: Especificar e implementar solo lo necesario de los TDAs que sean requeridos para abordar el problema.

Nota 3: Documentar y estructurar programa considerando: dominios, predicados, metas y cláusulas.

Pregunta 1 (30 pts)

El siguiente esquema representa los estados (A, B, C, X, Y y Z) de un sistema de microcontroladores que vigila la producción de chips de silicio a medida. Este esquema incluye el costo energético (en microwatts) para pasar de un estado a otro (por ejemplo, para ir del estado B al Y el sistema consume 60 microwatts).



Se le solicita crear en pseudo-Prolog un programa que permita planificar las operaciones de este sistema, en base a los siguientes requerimientos:

1. Expresar en una base de conocimiento de Prolog las relaciones del esquema anterior, indicando el identificador del estado, el costo energético y la dirección. Escoja un nombre de predicado acorde al contexto del problema. **(6 pto)**
2. Implementar un predicado `verificar(Origen, Destino, Costo)` que permita consultar si se puede ir de un estado a otro, o bien, a qué estado se puede llegar dado un costo determinado. **(8 pto)**
   Por ejemplo:
   `verificar(“A”, “X”, 50). => false`
   `verificar(“B”, “Z”, 70). => true`
   `verificar(“C”, ED, 95). => ED = “Z”`
3. Implementar un predicado `camino(Origen, Destino, Intermedio, Costo)` que calcule el costo total de todos los posibles caminos que conectan a dos estados, pero pasando por un punto intermedio (no necesariamente tienen que ser contiguos el origen, destino y el punto intermedio). **(8 pto)**
   Por ejemplo para ir de “A” a “X, pero pasando en algún momento por “C”:
   `camino(“A”, “X”, “C”, C). => C = 175; C = 85; C = 185.`

   Otro ejemplo: para ir de “A” a “X”, pero pasando en algún momento por “Z”, no es posible.
   `camino(“A”, “X”, “Z”, C). => false.`
4. Implementar un predicado `estadosIntermedios(Origen, Destino, Costo)` que retorne una lista con todos los estados intermedios de los posibles caminos que conectan a dos estados (no necesariamente tienen que ser contiguos). **(8 pto)**
   Por ejemplo:
   `estadosIntermedios(“A”, “X”, L). => L = [“B”]; L = [“B”, ”C”, “Y”];`
   `L = [“B”; ”Y”]; L = [“C”, “Y”];`

NOTA: Solo para esta pregunta, puede usar los predicados `member(Elemento, Lista)` que devuelve true si el Elemento está dentro de la Lista; y `append(L1, L2, LF)` que devuelve en LF la concatenación de las listas L1 y L2.

Pauta Pregunta 1:
NOTA: Pauta de referencia, probada parcialmente en SWI-Prolog

1)
```
transicion(a,b,50).
transicion(a,c,75).
transicion(b,c,15).
transicion(b,y,60).
transicion(c,y,85).
transicion(b,x,35).
transicion(y,x,25).
transicion(x,z,40).
transicion(y,z,10).
```

2)
```
verificar(Origen, Origen, 0).
verificar(Origen, Destino, Costo):-
  transicion(Origen, X, Y), C2 is (Costo - Y), verificar(X, Destino, C2).
```

3.1)
```
camino(O, O, I, 0).
camino(O, D, O, CT):-
  transicion(O, X1, C1), camino(X1, D, O, C2), CT is C1 + C2.
camino(O, D, I, CT):-
  transicion(O, X1, C1), camino(X1, D, I, C2), CT is C1 + C2.
```

3.2)
```
camino(O, D, I, CT):-
  verificar(O, I, C1), verificar(I, D, C2), CT is C1 + C2.
```

4)
```
estadosIntermedios(O, D, L):-
  transicion(O, D, _), L = [].
estadosIntermedios(O, D, L):-
  transicion(O, X, _), X \= D, estadosIntermedios(X, D, L2),
  \+(member(L2, X)), append([X], L2, L).
```

Pregunta 2 (30 pts)

El supermercado “El Económico” ofrece una gran variedad de productos. De los productos se tiene un registro detallado que incluye datos generales e información nutricional. Un ejemplo de los datos que se registran se presentan a continuación (los valores en las celdas son meramente de ejemplo y no representan la realidad),:

Código	Nombre	Tipo	Azúcares	Grasas	Sodio	Calorías	Precio	Stock
1	Helado	Postre	10	20	30	40	50	60
2	Yogurth	Lácteo	70	80	90	100	110	120
3	Queso	Lácteo	110	100	90	80	70	60
4	Donut	Confite	50	40	30	20	10	0

El área de Investigación, Desarrollo e Innovación del supermercado quiere habilitar un portal de compras en línea que permita a sus clientes armar carritos de compra saludables. En un primer prototipo del nuevo sistema (prueba de concepto) se ha optado por realizar su implementación en Pseudo-Prolog para lo cual le encargan a usted implementar un programa que cubra los siguientes aspectos:

a) Defina un predicado para representar productos según se ilustra en la tabla anterior e implemente las cláusulas necesarias para cada uno de los ejemplos de la tabla. Esta representación e implementación se debe tomar como base para el desarrollo de las siguientes preguntas **(5 pts)**

```
%dominios

%Codigo, Azucares,Grasas,Sodio,Calorias,Precio,Stock: Numero;

%Nombre,Tipo:String

%predicados

%producto(Codigo,Nombre,Tipo,Azucares,Grasas,Sodio,Calorias,Precio,Stock).

%metas

%producto

%clausulas

%hechos

producto(1,”Helado”,”Postre”,10,20,30,40,50,60).

producto(2, “Yogurth”,”Lácteo”,70,80,90,100,110,120).

producto(3,”Queso”,”Lácteo”,110,120,130,140,150,160).

producto(4,”Donut”,”Confite”,170,180,190,200,210,220).
```

b) total(ListaProductos,TotalCalorías,TotalGrasas,TotalAzucares,TotalSodio,TotalPrecio)

El cual permite hacer un cálculo del total de calorías, grasas, azúcares, sodio y precio en base al código de los artículos que se encuentran en ListaProductos, procurando verificar su existencia en la base de conocimiento **(5 pts)**.

ej: total([2,3,19,35,87],Calorías,Grasas,Azúcares,Sodio,Precio)

```
%dominios
%ListaProductos:Lista de Números
%TotalCalorías,TotalGrasas,TotalAzucares,TotalSodio,TotalPrecio: Número
%predicado
%total(ListaProductos,TotalCalorías,TotalGrasas,TotalAzucares,TotalSodio,TotalPrecio)
%metas
%total
%clausulas
%reglas
total([Cod],TC,TG,TA,TS,TP):-producto(Cod,_,_,TC,TG,TA,TS,_;TP).
total([Cod|Cods],TC,TG,TA,TS,TP):-producto(Cods,_,_,C,G,A,S,_,P),
                                  total(Cs,TC2,TG2,TA2,TS2,TP2),
                                  TC is C + TC2, TG is G + TG2, TA is A + TA2,
                                  TS is S + TS2, TP is P + TP2.
```

De los siguientes 2 items, solo responda 1.

c) carritoSaludable(ListaTiposCantidades,MaxCalorías,MaxGrasas,MaxAzucares,MaxSodio,MaxPrecio,ListaProductos):

El cual permite encontrar listados de productos (solo señalando sus códigos), en base a los tipos y cantidades indicados en ListaTiposCantidades (lista de pares de la forma [tipo,cantidad]). La lista resultante (ListaProductos) debe tener cómo máximo la cantidad de productos señalada para cada tipo en ListaTiposCantidades y mínimo 1 producto para cada tipo. Por otro lado, la lista de productos resultantes (totalizada) no debe exceder los valores nutricionales establecidos en MaxCalorías, MaxAzucares, MaxGrasas y, MaxSodio, además de no sobrepasar el presupuesto indicado en MaxPrecio. Se debe cautelar que la solución propuesta considere la existencia de productos (stock) **(20 pts).**

ej:

carritoSaludable([[lacteo,3],[confite,2],[postre,1],[carne,2]],1000,200,200,100,20000,LP).

Solución

Input:

carritoSaludable([[lacteo,3],[confite,2],[postre,1],[carne,2]],1000,200,200,100,20000,LP).

Ejemplo de Output:

ListaProductos = [1,3,5,4,2] // Lista código de productos filtrado de acuerdo a maxs y ListaTipoCantidades

```
% Caso base de recursión

carritoSaludable([],_,_,_,_,_,ListaProductos):-

        ListaProductos = [].


carritoSaludable([ [Tipo,CantidadMax]|RestoTipoCantidad],

                  MaxC,MaxG,MaxA,MaxS,MaxP,ListaProductos) :-

        producto(Código,_,Tipo,_,_,_,_,_,Stock),

        % Verificar que los totales no excedan los valores señalados

        total(TotalC, TotalG, TotalA, TotalS, TotalP, ListaProductos),

        TotalC=<MaxC, TotalG=<MaxG, TotalA=<MaxA, TotalS=<MaxS,TotalP=<MaxP,

        CantidadActual =< Stock,

    carritoSaludable(RestoTipoCantidad,MaxC,MaxG,MaxA,MaxS,MaxP,
RestoListaProductos),

    ListaProductos = [[Codigo,CantidadActual]|RestoListaProductos].
```

```
carritoSaludable([[Tipo|CantidadActual]],MaxC,MaxG,MaxA,MaxS,MaxP,NvaCola):-
        carritoSaludable(Tipo,CantidadActual,MaxC,MaxG,MaxA,MaxS,MaxP,NvaaCola).
```

- **El desarrollo de la pregunta se evalúa como: 0, 0.25, 0.5, 0.75, 1. Según el grado de cumplimiento de la respuesta entregada por el estudiante**
- **Cada feature o caracteristica de la solución puede ser implementada en cualquier orden y el cumplimiento de cada una de ellas es evaluado con un 0.25 acumulativo**

Feature	Evaluación
Si solo verifica que los totales no excedan los max	0,25
Hace verificación por código y tipo	0,5
Verifica por stock	0,75
Si descarta productos	1,0

d) mejorarCarrito(ListaProductosIn,MaxCalorías,MaxGrasas,MaxAzucares,MaxSodio,MaxPrecio,ListaProductosOut)

Permite realizar un análisis de un carrito expresado por ListaProductosIn (que contiene sólo los códigos de los productos) para verificar que los totales no exceda los valores señalados por MaxCalorías, MaxGrasas, MaxAzucares, MaxPrecio. Si los excede, entonces se busca una lista de productos alternativa (ListaProductosOut) a partir del descarte de los productos contenidos en ListaProductosIn hasta que los totales no excedan los máximos **(20 pts).**

ej:

mejorarCarrito([1,2,3,1,1,4,5,5,3,6,10],1000,200,200,100,20000,LP).

Solución

```
        % Caso base de recursión

        mejorarCarrito([], _, _, _, _, _, []).

        %Paso 1. Verificar que los totales no excedan los valores señalados (0,5)

        mejorarCarrito(ListaProductosIn,MaxC, MaxG, MaxA, MaxS, MaxP,
ListaProductosOut):-

                % A traves de unificación se obtienen los valores totales de cada
        propiedad señalada

                total(TotalC, TotalG, TotalA, TotalS, TotalP, ListaProductosIn),

                % Verificar que los totales no excedan los valores señalados

                TotalC=<MaxC, TotalG=<MaxG, TotalA=<MaxA, TotalS=<MaxS,TotalP=<MaxP.
```

```
        %Paso 2. Descartar productos de ListaProductosIn para generar
ListaProductosOut   (1.0)

        mejorarCarrito([_|Cola],MaxC,MaxG,MaxA,MaxS,MaxP,NvaCola):-

                mejorarCarrito(Cola,MaxC,MaxG,MaxA,MaxS,MaxP,NvaCola).
```

- **El desarrollo de la pregunta se evalúa como: 0, 0.25, 0.5, 0.75, 1. Según el grado de cumplimiento de la respuesta entregada por el estudiante**

Caso implementado	Evaluación
Si solo verifica que los totales no excedan el max	0,5
Si descarta productos	1,0