Apellido y Nombres:……. Legajo:..... Máquina:...

**Recuperatorio del Segundo Parcial de Paradigmas de Programación**

## Objetivo

Evaluar al estudiante en la parte práctica de las unidades nro. 5 y nro. 6 (Paradigma Funcional y Paradigma Lógico, respectivamente) a partir de la resolución que guarde en los archivos más abajo especificados, correspondientes a las consignas solicitadas para los ejercicios de cada paradigma.

## Condiciones de trabajo:

- Este parcial práctico consta de dos partes: programación lógica y programación funcional. Para cada paradigma se deberá desarrollar un programa, utilizando el correspondiente entorno.

- Para resolver las consignas correspondientes al paradigma lógico, se deberá generar un archivo con el nombre Legajo_ApellidoNombre.pl para definir los hechos y reglas, conforme se solicite en las consignas que se detallan más abajo.

- Para resolver las consignas correspondientes al paradigma funcional, se deberá generar un archivo con el nombre Legajo_ApellidoNombre.hs para formular las funciones que más abajo serán solicitadas.

- **Es responsabilidad de cada alumno ir guardando periódicamente cada archivo solicitado, como así también del contenido de los mismos, teniendo la precaución de guardarlo en el disco D: para su posterior backup.**

- En caso de que máquina no funcione correctamente durante el transcurso de la evaluación, debe notificar de esta situación a cualquier docente de la mesa examinadora.

- En ningún caso debe reiniciar la máquina, ya que perderá la totalidad del examen.
- El tiempo previsto para la realización de este examen es de **1:30 hs**.

# Paradigma Lógico

## Enunciado

Un mini mercado de barrio, nos solicitó implementar un programa en Prolog que ayude a la gestión de los envíos de los pedidos que solicitan los clientes.

A continuación se muestra la **tabla 1** que contiene los datos de los envíos a realizar en la jornada.

**Tabla 1: Pedidos programados.**

| Código de pedido | Estado | Repartidor asignado | Cliente | | | Carrito de ítems |
|---|---|---|---|---|---|---|
| | | | Nro. teléfono | Domicilio de entrega | | |
| | | | | Calle | Altura | |
| 1 | 4 | 'Lara' | 11111 | 'Salta' | 250 | ['A', 'A', 'D'] |
| 2 | 2 | 'Tina' | 22222 | 'Av. Colón' | 101 | ['B', 'C'] |
| 3 | 3 | 'Lali' | 44444 | 'San Martín' | 1321 | ['A', 'D'] |
| 4 | 1 | 'Tina' | 33333 | 'Avellaneda' | 3213 | ['B', 'D', 'D'] |
| 5 | 4 | 'Lara' | 99999 | '9 de Julio' | 2123 | ['A', 'B', 'B', 'B', 'E'] |
| 6 | 4 | 'Tina' | 88888 | 'Urquiza' | 1503 | ['D', 'D'] |

La **tabla 2** corresponde a los estados en los cuales puede estar un pedido.

**Tabla 2: Estados.**

| Código | Descripción |
|---|---|
| 1 | 'Registrado' |
| 2 | 'En preparación' |
| 3 | 'En camino' |
| 4 | 'Entregado' |
| 5 | 'Cancelado' |

La **tabla 3** corresponde al listado de ítems que puede seleccionar y agregar un cliente al carrito de ítems de su pedido.

**Tabla 3: Listado de ítems.**

| Código | Descripción | Precio [$] |
|---|---|---|
| 'A' | 'Pack 1' | 2500 |
| 'B' | 'Pack 2' | 1900 |
| 'C' | 'Pack 3' | 2200 |
| 'D' | 'Pack 4' | 2800 |
| 'E' | 'Pack 5' | 2000 |

## Su tarea:

A partir de los hechos ya definidos que representan todos los datos de las tablas 1, 2 y 3, usted deberá definir las reglas que permitan resolver lo siguiente:

1) Conocer la descripción del estado de pedido, la calle y altura del domicilio de entrega, de todos los pedidos programados cuyo nombre de repartidor se especifica. Predicado sugerido para esta regla: regla1/4. **(15 puntos)**

*Por ejemplo:*

**regla1('Lara',Estado,Calle,Altura).**
Estado = 'Entregado',
Calle = 'Salta',
Altura = 250;
Estado = 'Entregado',
Calle = '9 de Julio',
Altura = 2123.

**regla1('Tina',Estado,Calle,Altura).**
Estado = 'En preparación',
Calle = 'Av. Colón',
Altura = 101;
Estado = 'Registrado',
Calle = 'Avellaneda',
Altura = 3213;
Estado = 'Entregado',
Calle = 'Urquiza',
Altura = 1503.

**2)** Conocer si existe o no existe algún pedido que se encuentre en estado cancelado o en estado registrado para cierto repartidor asignado cuyo nombre se especifica. Nombre de la regla: regla2/1. **(15 puntos)**

*Por ejemplo:*

**regla2('Lara').**
false.

**regla2('Tina').**
true.

**3)** Generar una lista con los códigos de aquellos pedidos que incluyan en su carrito algún ítem cuya descripción se especifica. Nombre de la regla: regla3/2. **(20 puntos)**

*Por ejemplo:*

**regla3('Pack 1', Lista).**

Lista = [1, 3, 5].

**regla3('Pack 2', Lista).**

Lista = [2, 4, 5].

**regla3('Pack 3', Lista).**

Lista = [2].

## Paradigma Funcional

**Enunciado**

Continuando con el caso de estudio anterior, el mini mercado también requiere que se le implementen un conjunto de funciones en Haskell.

**1)** Realizar una función que reciba un código de ítem y devuelva el precio del mismo, de acuerdo a la tabla 4. En caso que el código de tipo de ítem no sea ninguno de los especificados en la tabla 4, la función deberá retornar cero. **(15 puntos)**

**Tabla 4: Listado de ítems.**

| Código | Precio [$] |
|---|---|
| 'A' | 2500 |
| 'B' | 1900 |
| 'C' | 2200 |
| 'D' | 2800 |
| 'E' | 2000 |

Ejemplo: **funcion1 'B'** deberá devolver, **1900**.
Ejemplo: **funcion1 'E'** deberá devolver, **2000**.
Ejemplo: **funcion1 'F'** deberá devolver, **0**.

**2)** Realizar una función que reciba un código de ítem como parámetro, y devuelva True en caso que dicho código se corresponda con alguno de los códigos de ítems de la tabla 4, o que devuelva un False, en caso que el código recibido por parámetro no se corresponda con ninguno de los códigos de ítems de la tabla 4.

Ejemplo: **funcion2 'B'** deberá devolver, **True**.
Ejemplo: **funcion2 'E'** deberá devolver, **True**.
Ejemplo: **funcion2 'F'** deberá devolver, **False**.

**3)** Realizar una función, que a partir de un código de ítem que se recibe como primer, y de una lista de códigos de ítems que se reciben como segundo parámetro, devuelva la cantidad de veces que el código de ítem correspondiente al primer parámetro, se encuentra en la lista de ítems correspondiente al segundo parámetro. **Utilizar recursividad**. **(15 puntos)**

Ejemplo: **funcion3 'A' ['A', 'E', 'A']** deberá devolver, **2**. El código de ítem **'A'** se encuentra 2 veces en la lista.
Ejemplo: **funcion3 'C' ['A', 'E', 'A']** deberá devolver, **0**. El código de ítem **'C'** se encuentra 0 veces en la lista.