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

**Segundo Parcial de Paradigmas de Programación Tema A**

**Objetivo**

Evaluar al estudiante en la parte práctica de las unidades nro. 5 y nro. 6 (Paradigma Lógico y Paradigma Funcional, 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\_AplellidoNombre.pl para definir los hechos y reglas, conforme se solicite en las consignas que se detallan más abajo. También se deberá generar un archivo con el nombre Legajo\_ApellidoNombre.txt para formular los objetivos solicitados más abajo y su correspondiente respuesta de Prolog.
* Para resolver las consignas correspondiente al paradigma funcional, se deberá generar un archivo con el nombre Legajo\_AplellidoNombre.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.
* **Es responsabilidad de cada estudiante el contenido que quede guardado al momento de finalizar la evaluación, en todos sus archivos generados durante el examen.**
* El **tiempo previsto** para la realización de este parcial es **1 hs más 30 minutos**.

**Evaluación de Programación Lógica**

**Caso de estudio:**

Un restaurante criollo necesita un programa en Prolog para administrar los pedidos de las diferentes mesas. A continuación se muestran algunos pedidos realizados y los correspondientes precios de cada plato.

**Tabla 1: Pedidos.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Código de pedido** | **Mesa** | **fecha** | | | **Mesero** | **Platos** |
| **dd** | **mm** | **aaaa** |
| c1 | 4 | 30 | 10 | 2016 | Pedro | [‘pizza’, ‘empanada’, ‘empanada’] |
| c2 | 3 | 30 | 10 | 2016 | Laura | [‘locro’, ‘locro’, ‘empanada’] |
| c3 | 4 | 30 | 10 | 2016 | Lisa | [‘tamal’, ‘empanada’] |
| c4 | 2 | 2 | 11 | 2016 | Lisa | [‘flan’, ‘tamal’, ‘tamal’] |
| c5 | 3 | 2 | 11 | 2016 | Fiamma | [‘locro’, ‘locro’, ‘tamal’, ‘empanada’, ‘flan’] |
| c6 | 1 | 2 | 11 | 2016 | Fiamma | [‘empanada’, ‘empanada’, ‘empanada’, ‘empanada’, ‘flan’] |

**Tabla 2: Precios por plato.**

|  |  |  |
| --- | --- | --- |
| **Código de plato** | **Plato** | **Precio [$]** |
| 1 | pizza | 75 |
| 2 | empanada | 20 |
| 3 | locro | 80 |
| 4 | tamal | 30 |
| 5 | flan | 40 |

**Su tarea:**

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

1. Retornar una lista con los nombres de aquellos meseros que atendieron pedidos durante el mes 11 del año 2016. Si un mesero atendió varios pedidos durante este mes, su nombre sólo deberá aparecer una sola vez. Predicado sugerido para esta regla: regla1/1. **(15 puntos)**

**regla1(L1):- findall(N,(pedidos(\_,\_,fecha(\_,M,A),N,\_),M=:=11,A=:=2016),L),sort(L,L1).**

1. ¿Cuantas mesas fueron atendidas por un mesero determinado?Predicado sugerido para esta regla: regla2/2. **(15 puntos)**

**regla2(Mesero,C):- findall(Mesa,(pedido(\_,Mesa,\_,Mesero,\_),L),length(L,C).**

1. Dado un código de plato mostrar el listado de los códigos de pedidos que posean ese plato junto con el precio del mismo. Predicado sugerido para esta regla: regla3/3. **(20 puntos)**

**regla3Aux(CPlato,CPedido,Precio):- preciosporplato(CPlato,NomPlato,Precio),pedido(CPedido,\_,\_,\_,LPL), member(NomPlato,LPL).**

**regla3(CPlato,ListaPedidos):-findall([CPedido,Precio], regla3Aux(CPlato,CPedido,Precio), ListaPedidos)**

**Evaluación de Programación Funcional**

Continuando con el caso de estudio anterior, realizar las siguientes funciones en Haskell:

**Su tarea:**

**Realiza las funciones que permitan resolver lo siguiente:**

1. **precio\_plato (plato)**: que a partir del nombre del plato retorne el precio individual correspondiente según la tabla 2. En caso de recibirse un nombre de plato que no figure en la tabla, deberá devolver 0. **(15 puntos)**

Ejemplo:

**precio\_plato** (‘empanada’)

Resultado: 20.

precio\_plato(plato) |(plato==’pizza’)=75

|(plato==’empanada’)=20

|(plato==’locro’)=80

|(plato==’tamal’)= 30

|(plato==’flan’)=40

|otherwise=0

1. **precio\_pedido (lista)**: que a partir de una lista de platos correspondientes a un pedido, calcule y retorne el precio total del pedido (es decir, la suma de los precios individuales de cada plato que figura en lista). **(15 puntos)**

Ejemplo:

**precio\_pedido** ([‘empanada’, ‘tamal’, ‘empanada’, ‘tamal’, ‘empanada’])

Resultado: 120.

precio\_pedido([])=0

precio\_pedido(x:xs)= precio\_plato(x) + precio\_pedido(xs).

1. **generar\_lista(lista):** que a partir de una lista de precios de comidas genere otra lista con aquellos precios que superen un valor p recibido como argumento. **(20 puntos)**

Ejemplo: **generar\_lista** ([20, 40, 60, 40]), si el valor p=40.

Resultado: [60].

generar\_lista [] \_=[]

generar\_lista (x:xs) p= if x > p then x: generarLista xs p else generarLista xs p