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

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. **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 correspondientes al paradigma funcional, se deberá generar un archivo con el nombre Legajo_ApellidoNombre.hs para formular las funciones que más abajo serán solicitadas. **También deberá generar un archivo con el nombre Legajo_ApellidoNombre.txt para formular los requerimientos solicitados más abajo y su correspondiente resultado de Haskell (Esto será considerado en la evaluación).**

- **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

Una consultora de recursos humanos necesita un programa en Prolog que le brinde cierta información sobre las diferentes evaluaciones tomadas a los aspirantes para ciertos puestos de trabajo de una aerolínea.

A continuación se muestra la **tabla 1** que contiene los datos de los aspirantes registrados.

Tabla 1: Aspirantes

DNI	Nombre	Apellido	Nro. teléfono	Domicilio	
				Calle	Altura
'30000000'	'Lara'	'Pointer'	11111	'Suipacha'	250
'35000000'	'Tina'	'Jolie'	22222	'San Martín'	101
'25000000'	'Paty'	'Pérez'	44444	'Rivadavia'	1321
'20000000'	'Matilde'	'Smith'	99999	'9 de Julio'	2123

La **tabla 2** corresponde a las diferentes evaluaciones realizadas a cada uno de los aspirantes, para diferentes ítems. La selección de qué ítems que se evaluaron en cada una, dependieron de factores aleatorios.

Tabla 2: Evaluaciones de los aspirantes

Código de evaluación	DNI del aspirante	Fecha de evaluación			Ítems evaluados	Puntajes obtenidos
		Día	Mes	Año		
1	'30000000'	12	5	2019	['A', 'B', 'C']	[10, 15, 20]
2	'20000000'	12	5	2019	['B, 'C']	[12,15]
3	'30000000'	19	5	2019	['D', 'E']	[19, 18]
4	'20000000'	19	5	2019	['A', 'D', 'E']	[6, 15, 12]
5	'30000000'	1	6	2019	['F']	[15]

La **tabla 3** corresponde a los diferentes ítems que se tuvieron en cuenta en las evaluaciones, y el puntaje máximo para su ponderación.

Tabla 3: Ítems a evaluar

Código del ítem	Descripción	Puntaje máximo
'A'	Uniforme.	10
'B'	Trabajo colaborativo.	15
'C'	Cumplimiento de tareas.	20
'D'	Compañerismo.	20
'E'	Idioma inglés.	20
'F'	Idioma portugués.	15

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 si entre los ítems a evaluar, existe algún ítem cuyo puntaje máximo sea menor o igual al primer parámetro, o si existe algún ítem cuyo puntaje máximo sea mayor o igual al segundo parámetro. Predicado sugerido para esta regla: regla1/2. **(15 puntos)**

   Por ejemplo:

   regla1(5,25).

   false

   regla1(10,25).

   true

2) Conocer para cierta evaluación realizada cuyo código se deberá especificar, los siguientes otros datos: apellido y nombre del aspirante evaluado, calle y altura de su domicilio. Predicado sugerido para esta regla: regla2/5. **(15 puntos)**

   Por ejemplo:

   regla2(1,Ape,Nom,Calle,Altura).

   Ape='Pointer'

   Nom='Lara'

   Calle='Suipacha'

   Altura=250

3) Generar una lista con los DNI de aquellos aspirantes que tuvieron alguna evaluación durante el mes y año que se especifique. Considerar la posibilidad que un mismo aspirante haya sido evaluado más de una vez en ese mes y año, por lo que se deberá evitar que su DNI figure repetido tantas veces como se lo haya evaluado en ese mismo mes y año.
   Predicado sugerido para esta regla: regla3/3. **(20 puntos)**

   Por ejemplo:

   regla3(5,2019,Lista).

   Lista=['30000000', '20000000'].

   regla3(6,2019,Lista).

   Lista=['30000000'].

Paradigma Funcional

Enunciado

La misma consultora de recursos humanos nos ha solicitado además el desarrollo de un programa en Haskell que permita cumplir los siguientes requerimientos.

1) Realizar una función que reciba el puntaje obtenido en la prueba de suficiencia para un aspirante y retorne la categoría resultante. En el caso de que el puntaje no se encuentre ninguno de los rangos especificados en la Tabla 1 la función deberá retornar un mensaje de "Puntaje erróneo". **(15 puntos)**

Tabla 1. Categoría del Aspirante

Rango Puntaje	Categoría
90 a 100	"Nivel Muy Bueno"
75 a 89	"Nivel Bueno"
60 a 74	"Nivel Aceptable"
45 a 59	"Nivel Regular"
0 a 44	"Nivel Inaceptable"

Provista la siguiente lista de enteros, en la que cada elemento de la misma representa el puntaje obtenido de los aspirantes, resolver los requerimientos solicitados en las consignas 2 y 3.

```
lista :: [Integer]
lista = [60, 80, 42, 95, 100, 89]
```

2) Realizar una función que genere una nueva lista con los puntajes que pertenezcan a un determinado rango. Considerar como parámetros de la función: la lista con los puntajes de todos los aspirantes, el rango inferior y superior. **(15 puntos)**

Ejemplo: [60, 80, 42, 95, 100, 89] 90 100, el resultado esperado sería: **[95, 100]**

3) Realizar una función que permita retornar la cantidad de puntajes que sean superior a un valor "x". Considerar como parámetros de la función: la lista con los puntajes de todos los aspirantes y el valor "x". **Para resolver este punto debe definir una función recursiva**. **(20 puntos)**

Ejemplo: [60, 80, 42, 95, 100,89] x=50, el resultado esperado sería: **5**