Pontificia Universidad Javeriana
Conceptos y Modelos de Programación

Taller de Programación Declarativa

¨El estudiante de la Pontificia Universidad Javeriana, como agente de su propia formación, es corresponsable de la Identidad Institucional, uno de cuyos cimientos es tener como hábito un comportamiento ético en todos los ámbitos de la vida. En este sentido me comprometo a realizar con total integridad esta evaluación, solamente empleando los recursos autorizados para su desarrollo¨.
Consejo Académico, Acta Nro 79, abril 19 de 2004

Pregunta	1	2	3	4	Total
Puntos	25	25	25	25	100
Cal.					

1 (25 Puntos)
La función que se presenta a continuación calcula el n-ésimo elemento de la serie de *fibonacci*.

```
fun {Fibonacci N}
   case N
   of 0 then 1
   [] 1 then 1
   else
      {Fibonacci N-2} + {Fibonacci N-1}
   end
end
```

Teniendo en cuenta la implementación anterior:

(a) Defina una cota para la complejidad temporal y espacial. Para la temporal comience por definir una ecuación de recurrencia. La complejidad espacial la puede definir en términos del espacio que ocupa la función en la memoria del interprete o de la máquina abstracta que utiliza la ejecución del programa.

(b) Escriba una versión diferente de la función anterior que cumpla las siguientes condiciones:

- Complejidad temporal lineal
- Complejidad espacial constante

Para lo anterior, encuentre una nueva ecuación de recurrencia que le permita deducir la complejidad lineal. Además explique por qué la nueva función tiene complejidad espacial constante.

(c) Escriba un invariante para la función del ejercicio anterior y muestre que ambas funciones, la primera y la nueva, son equivalentes.

2 (25 Puntos)

(a) Escriba una función *Filter* que tome como argumentos una función unaria y una lista y retorne una lista con todos los elementos par los cuales la función retornó verdadero. La función propuesta debe ser eficiente tanto en tiempo ($O(n)$) como en espacio (constante).

(b) A continuación se muestra la implementación de las funciones *MFoldL* y *MFoldR*.

```
fun {MFoldL F L M}
   case L
   of nil then M
   [] H|T then {MFoldL F T {F H M}}
   end
end

fun {MFoldR F L M}
   case L
   of nil then M
   [] H|T then {F H M} | {MFoldL F T M}
   end
end
```

(a) Explique cuáles son las principales diferencias entre ambas funciones. Tenga en cuenta su complejidad.

(b) Cree usted que las dos funciones son equivalentes? (Explique su respuesta).

(c) Utilizando cualquiera de las dos funciones anteriores, defina las funciones *Mayor* y *Menor* que retornan el mayor (resp. menor) elemento de una lista.

(d) Redefina las funciones *Map* y *Filter* en términos de *MFoldL*.

3 (25 Puntos)
Escriba una función que implemente el método de *Newton-Raphson*. Utilice solamente las técnicas vistas en el curso.

4 (25 Puntos)

(a) Escriba una función que dadas dos listas de números ordenadas de menor a mayor retorne una nueva lista con los números de ambas de manera ordenada. Llame a la función *Mezclar*.

(b) Utilice la función anterior para crea una función que dada una lista de números los ordene de menor a mayor.

(c) Cambie las funciones para que ordenen listas de cualquier clase de elementos. Para esto, la función recibirá un parámetro adicional que es una función que define el ordenamiento deseado. Por ejemplo, si la función es **fun** {$ X Y} X $<$ Y **end**, la lista se ordenará de menor a mayor; si la función es **fun** {$ X Y} X $>$ Y **end**, la lista se ordenará de mayor a menor.