Universidad de Puerto Rico en Bayamón

Departamento de Ciencias de Computadoras

COTI039 – Comparación de Lenguajes de Programación

Asignación #4 – 75 puntos

**NOTA: No se permite el uso de atributos del lenguaje de programación que no hayan sido discutidos en clase, asignados para lectura o autorizados explícitamente.**

**Ejercicios:**

1. (15 puntos) Escriba un programa en F# (fibonacci.fs) que calcule el enésimo elemento de la sucesión de Fibonacci. A continuación se muestra un ejemplo de su ejecución:

> start 7;;

Using a tail-recursive helper function:

The 7-th number in the Fibonacci sequence is 13

Using a pattern matching function:

The 7-th number in the Fibonacci sequence is 13

Using a pattern matching expression:

The 7-th number in the Fibonacci sequence is 13

Using an if expression:

The 7-th number in the Fibonacci sequence is 13

Si el número es menor de 0, el programa deberá mostrar un mensaje de error y terminar. La versión que usa recursión regular puede explotar si se tratan valores mucho mayores de 30. El programa deberá determinar el elemento de cuatro formas:

* Mediante una función auxiliar que use recursión de cola.
* Mediante una función de pareo de patrones.
* Mediante una expresión de pareo de patrones.
* Mediante una expresión if.

2. (15 puntos) Escriba un programa en F# (scalar\_mult.fs) que determine la multiplicación escalar de un número entero y una lista de números enteros. A continuación se muestra un ejemplo de su ejecución:

> start 3 [2;7;4];;

Using the List.map function:

The scalar multiplication is [6;21;12]

Using a tail-recursive helper function:

The scalar multiplication is [6;21;12]

Using regular recursion:

The scalar multiplication is [6;21;12]

Si la lista está vacía, el programa deberá mostrar un mensaje de error y terminar. El programa deberá determinar el producto escalar de tres formas:

* Mediante la función *map* del módulo *List* y una función anónima que multiplique el escalar por el elemento examinado.
* Mediante una función auxiliar que use recursión de cola.
* Mediante recursión regular.

Nota: Para resolver este problema, permitiré que se utilice la función *rev* del módulo *List*.

3. (15 puntos) Escriba un programa en F# (list\_min.fs) que determine el elemento mínimo en una lista de números enteros. A continuación se muestra un ejemplo de su ejecución:

> start [5;10;4;3;2;9;1;6;-7;8];;

Using the List.min function:

The minimum element is -7

Using the List.fold function:

The minimum element is -7

Using a tail-recursive helper function:

The minimum element is -7

Using regular recursion:

The minimum element is -7

Si la lista está vacía, el programa deberá mostrar un mensaje de error y terminar. El programa deberá determinar el mínimo de cuatro formas:

* Mediante la función *min* del módulo *List*.
* Mediante la función *fold* del módulo *List* y una función anónima que determine el menor entre el acumulador y el elemento examinado.
* Mediante una función auxiliar que use recursión de cola.
* Mediante recursión regular.

4. (15 puntos) Escriba un programa en F# (odd\_count.fs) que determine la cantidad de elementos impares en una lista de números enteros. A continuación se muestra un ejemplo de su ejecución:

> start [5;10;4;3;2;9;1;6;-7;8];;

Using the List.filter and List.length functions:

The number of odd elements is 5

Using a tail-recursive helper function:

The number of odd elements is 5

Using regular recursion:

The number of odd elements is 5

Si la lista está vacía, el programa deberá mostrar que hay 0 elementos impares. El programa deberá determinar cantidad de elementos impares de tres formas:

* Mediante la función *filter* del módulo *List* y una función anónima que determine si el elemento examinado es impar. Necesitará luego usar el operador de tubería (*pipe*) y la función *length* del módulo *List* para contar los impares.
* Mediante una función auxiliar que use recursión de cola.
* Mediante recursión regular.

5. (15 puntos) Escriba un programa en F# (ins\_sort.fs) que permite insertar y ordenar ascendentemente los elementos en una lista de números enteros. A continuación se muestra un ejemplo de su ejecución:

> insert 4 [];;

[4]

> insert 4 [1;2;3;5];;

[1;2;3;4;5]

> insertion\_sort [5;10;4;3;2;9;1;6;-7;8];;

[-7;1;2;3;4;5;6;8;9;10]

El programa deberá contener dos funciones:

* Una función *insert* que inserta un elemento en una lista ordenada y la mantiene ordenada.
* Una función *insertion\_sort* que utiliza *insert* para ir formando la lista ordenada.

**Notas:**

1. Documente sus programas colocando su nombre, número de estudiante, fecha de creación, el nombre del archivo fuente y una descripción general. Cada función deberá contener un comentario descriptivo que indique lo que hace, lo que recibe y lo que devuelve.
2. Envíe una carpeta comprimida con copia de los archivos fuentes a la dirección de correo electrónico de su profesor utilizando **COTI 4039-*xxx*, Asignación #4** como asunto (*subject*) del mensaje, donde *xxx* es la sección en la que está matriculado.