Ejercicios de Lenguajes – Paradigma Funcional

Ejercicios

**NOTAS:**

* **Los siguientes ejercicios pretenden que el programador se enfoque en el algoritmo que resuelve el problema y no en la interacción con el usuario para obtener datos, así que se pueden trabajar con datos quemados y haciendo pruebas de funciones/métodos controladas por el programador en el main de los programas si se requiere.**
* **Los ejercicios deben ser implementados por el/la estudiante y no utilizar otras funciones que existan que lo realicen ya en F# (si es que existe)**
* **Todos los ejercicios deben subirse al repositorio**

1. Realizar ejercicio para desplazar (SHIFT) una lista de elementos n posiciones a la izquierda o la derecha según se indique por argumento. Ej:

desplazar “izq” 3 [1,2,3,4,5]

[4,5,0,0,0]

desplazar “der” 2 [1,2,3,4,5]

[0,0,1,2,3]

desplazar “izq” 6 [1,2,3,4,5]

[0,0,0,0,0]

1. Haciendo uso de la función filter, implemente una función que, a partir de una lista de cadenas de parámetro, filtre aquellas que contengan una subcadena que el usuario indique en otro argumento. Ej

sub\_cadenas “la” [“la casa, “el perro”, “pintando la cerca”]

[“la casa, “pintando la cerca”]

1. Realizar una función que obtenga el n-esimo elemento de una lista pero utilizando solamente map (sin recursión). Tip: Utilizar lista de índices como segunda lista de argumento para el map.

n\_esimo 2 [1,2,3,4,5]

3

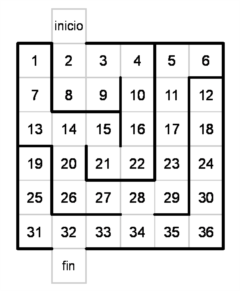
n\_esimo 3 [1,2,3,4,5]

4

n\_esimo 6 [1,2,3,4,5]

False

1. Modifique el ejercicio de búsqueda en profundidad visto en clase para ingresar pesos a los vértices (una recomendación es representarlos como [["i"],["a","b"],**[**"**3**"**,**"**6**"**,**"**5**"**]**] para evitar tener que cambiar algo en el algoritmo actual). Implemente una función que encuentre el camino más corto en la búsqueda en profundidad basándose en la sumatoria de pesos. Implica cambiar la búsqueda en profundidad para que se lleven los pesos además de los vértices.
2. Construya un programa que resuelva el problema de encontrar la o las rutas de un laberinto con datos quemados. Dicho problema se puede resolver exactamente igual que un problema de grafos, entendiendo que cada casilla del laberinto puede verse como un “vecino” adjunto de otra, siempre que no haya pared:



Así por ejemplo, en el laberinto anterior, se puede entender que por ejemplo, la posición 2 está conectada con la 3 y la 8 y en ese sentido y siguiendo el ejemplo de búsqueda en profundidad visto en clase, se puede representar como el vértice [[2],[3,8]] del eventual grafo que representa todo el laberinto.

* Represente el laberinto del gráfico como un grafo y corra la búsqueda en profundidad para encontrar todos los caminos que lo resuelven. Elimine paredes y represente ese nuevo conocimiento de caminos posibles en el grafo y vuelva a probar.
* Escriba una función que encuentre la ruta más corta para resolver el laberinto