PRACTICO Nº 11

Introducción

El objetivo de este práctico es ejercitar de forma introductoria los conceptos sobre estructuras de datos y entrada/salida vistos en el curso.

Ejercicio 1

a) Escriba una función *sensoresCriticos* que reciba un cell array *sensores* donde cada fila representa un sensor y las columnas 1 a 4 representan lo siguiente:
nombre (texto): nombre del sensor.
ubicacion (texto): ubicación en la planta.
valor (decimal): valor actual de lectura.
umbral (decimal): valor de umbral máximo permitido.

La función debe devolver un cell array *alarma* de dos columnas con los nombres y ubicaciones de los sensores cuyo valor supere el umbral.

b) Escriba la misma función, pero recibiendo un vector de structs (en lugar de un cell array) donde cada struct contiene los campos representados por las columnas del cell array anterior, y devolviendo también un vector de structs con los campos nombre y ubicación, representando los sensores que exceden el umbral.

c) Escriba una función *convCellAStructs* que reciba un cell array como *sensores* de la parte a) y lo convierta a un vector de structs como el que recibe la función de la parte b).

Ejercicio 2

a) Escriba una función *crearMatrizDispersa* que reciba tres vectores:
filas (vector de enteros): contiene los índices de fila de los elementos no nulos.
columnas (vector de enteros): contiene los índices de columna de los elementos no nulos.
valores (vector de decimales): contiene los valores correspondientes en esas posiciones.

La función debe devolver un struct con los siguientes campos:
cantFilas (entero): cantidad de filas de la matriz.
cantColumnas (entero): cantidad de columnas de la matriz.
noceros (vector de structs): cada entrada del vector contiene los campos *fil*, *col* y *val* con los índices de fila, columna y valor de un elemento distinto de cero de la matriz.

b) Escriba una función *promedioFilas* que reciba una matriz dispersa representada mediante un struct como el de la parte a), y un vector con un conjunto de índices de fila. La función devuelve una matriz de dos columnas cuya primera columna contiene los índices de fila de la entrada y la segunda contiene el promedio de los valores correspondientes a esas filas.

Ejercicio 3

a) Escriba una función *buscarPrimerMayor* que recibe:
- *cellArr:* un cell array donde cada fila tiene dos celdas: la primera numérica y la segunda de otro tipo.
- *val:* un valor numérico.

y devuelve el índice de la primera fila de *cellArr* donde la celda con valor numérico es mayor que *val.*

b) Escriba una función *insertarOrdenado* que reciba un cell array de tamaño 1x2 *nuevafila* con la primera celda numérica y la otra de otro tipo, y un cell array *cellArr* como el de la parte a). La función debe devolver el resultado de insertar *nuevafila* en *cellArr* de forma que las filas de *cellArr* se mantengan ordenadas de forma ascendente según el valor numérico de la primera columna.

c) Escriba una función *ordenarCA* que reciba un cell array cuya primera columna es numérica y devuelva el resultado de ordenar sus filas de forma ascendente según el valor de la primera columna.

Ejercicio 4

Escriba una función *poliVal* que reciba un vector de structs que representa un polinomio *P* (cada elemento es un struct con los campos *coef* y *grado*) y un número *x*. La función devuelve el resultado de evaluar *P(x)*.

Ejercicio 5

a) Escriba una función *distribuirCIs* que reciba un entero representando un número de cédula y un cell array de 10 elementos, donde cada elemento es un cell array de largo variable. La función debe agregar el número al cell array que corresponde con el último dígito de la cédula.

b) Escriba una función *estaCI* que reciba un entero representando un número de cédula y un cell array de 10 elementos como el de la parte anterior y devuelva 1 si la cédula se encuentra almacenada en la estructura y 0 en caso contrario.

Ejercicio 6

a) Escriba una función *buscarPalabra* que recibe un cell array de palabras (vectores de caracteres) y un vector de caracteres que representa una palabra. La función devuelve 1 si la palabra se encuentra en el cell array y 0 en caso contrario. Para comparar palabras utilice la función *strcmp* de Octave.

b) Escriba una función *agruparPalabras* que recibe un cell array de palabras (vectores de caracteres) y devuelve un cell array de 27 filas donde cada fila contiene dos celdas: la primera es una letra y la segunda es un cell array con la lista de palabras que comienzan con esa letra. Las filas del cell array de salida deben estar ordenadas alfabéticamente (asuma que las palabras comienzan en letras minúsculas entre a y z). No deben almacenarse palabras repetidas.

Ejercicio 7

Se representan puntos en el plano cartesiano como structs con los campos *x* e *y*. A su vez, un segmento de recta es un vector de dos elementos, donde cada uno es un punto. Escriba una función *esFiguraCerrada* que reciba una matriz de structs de tamaño *p* x 2, que representa un conjunto de *p* segmentos de recta y devuelva 1 si corresponde a una figura cerrada (cada punto aparece exactamente en 2 segmentos distintos del conjunto y no hay segmentos repetidos o superpuestos).

Ejercicio 8

a) Escriba una función recursiva *insertarEnArbol* que reciba un entero y un struct *arbol* con 3 campos (valor,menores,mayores). La función debe crear un struct con el número y dos structs vacíos e insertarlo en el primer struct vacío de *arbol* que se encuentra navegando por la rama correcta según el valor del entero.

Ejemplo:
arbol = struct('val',5,'mayores',[],'menores',struct('val',3,'mayores',[],'menores',[]))
arbol = insertarEnArbol(4,arbol) → arbol.menores.mayores.val == 4

b) Escriba una función recursiva *arbolAVector* que reciba un *arbol* como el de la parte anterior y devuelva un vector con los enteros que están almacenados ordenados de menor a mayor.

Ejemplo:
```
arbol = struct('val', 5,
               'mayores', [],
               'menores', struct( 'val', 3,
                                  'mayores', struct('val',4,
                                                    'mayores',[],
                                                    'menores',[]),
                                  'menores',[]))
```
arbolAVector(arbol) → [3,4,5]

Ejercicio 9

Escriba una función recursiva en Octave que reciba un cell arrray donde cada elemento es una matriz y devuelva 1 si todas las matrices almacenadas poseen la misma cantidad de filas y 0 en caso contrario.

Ejercicio 10

Considere una matriz de structs, donde cada elemento contiene una matriz dispersa en formato elemental almacenada como en el Ejercicio 2 parte a).

a) Escriba una función en Octave que reciba una matriz de structs *MT* como la anterior y devuelva un vector *maxFils* que contenga la cantidad de filas de la matriz dispersa con más filas en cada fila de *MT* y un vector *maxCols* que contenga la cantidad de columnas de la matriz dispersa con más columnas en cada columna de *MT*.

b) Escriba una función en Octave que reciba una matriz de structs *MT* y dos vectores maxFils y maxCols como los obtenidos en la parte a) y devuelva la matriz resultado de concatenar las matrices densas correspondientes a cada matriz dispersa almacenada en *MT*. El tamaño de la matriz densa resultante de expandir la matriz dispersa *MT(i,j)* debe ser *maxFils(i)* x *maxCols(j)*.

Ejercicio 11

Considere la siguiente estructura de datos para almacenar los archivos que se encuentran en una computadora. Cada carpeta o archivo se almacena mediante un struct con los siguientes campos:
nombre (vector de caracteres): nombre del archivo o carpeta.
tipo (0 o 1): 0 corresponde a un archivo y 1 corresponde a una carpeta.
contenido (vector de structs): cada entrada del vector contiene un struct representando los archivos y carpetas almacenados en esa ubicación.

a) Escriba una función recursiva *ls* que reciba un struct *dir* representando una carpeta y devuelva un cell array con los nombres de los archivos y carpetas contenidos en *dir*.

b) Escriba una función recursiva *ls2* que reciba un struct *dir* y un cell array *ruta* conteniendo la ruta para navegar desde la carpeta *dir* a otra carpeta. La función devuelve un cell array con los nombres de los archivos contenidos en la carpeta especificada por *ruta.* Si no es posible navegar desde la carpeta *dir* a la carpeta especificada por *ruta* la función debe devolver -1.

Ejemplo:
para la siguiente estructura de carpetas *carpeta1*:

carpeta1	→	carpeta2	→	Arch5.jpg		
		carpeta3	→	carpeta4	→	Arch3.txt
		Arch1.txt		Arch3.tar.gz		Arch4.mp4
		Arch2.pdf		Arch4.png		

ls2(carpeta1, {'carpeta1','carpeta3','carpeta4'})→ {'Arch3.txt','Arch4.mp4'}
ls2(carpeta1, {'carpeta3','carpeta4'})→ -1
ls2(carpeta2, {})→ {'Arch5.jpg'}
ls2([], {'carpeta3','carpeta4'})→ -1

Guía de solución:
Interpretar el parámetro dir como un vector de structs. En cada paso de la recursión tomar el primer elemento del vector y comparar el campo nombre con el primer elemento del cell array ruta. Si el primer elemento es una carpeta y el nombre coincide con el primer nombre del cell array ruta invocar a la función recursiva con el vector *contenido* de la carpeta, eliminando el primer elemento del cell array ruta. Si no coincide o no es un directorio invocar a la función recursiva con el resto del vector de structs. Si el cell array ruta es vacío la función devuelve el contenido de la carpeta actual. Si el vector de structs es vacío no es posible navegar esa ruta de carpetas. Puede utilizar la función de la parte a).