COMPUTACIÓN 1
Instituto de Computación

2do Parcial - Noviembre de 2009 - 2ª parte

- Duración del parcial: 2:45 Hs.
- No se podrá utilizar ningún tipo de material (apuntes, libro, calculadora, etc). Apague su teléfono celular.
- **Sólo** se contestarán preguntas sobre interpretación de la letra hasta 30 minutos antes de la finalización del mismo.
- Escriba las hojas de un solo lado
- Las partes no legibles del examen se considerarán no escritas
- En la primer hoja a entregar ponga con letra clara, en el ángulo superior derecho, salón en el cual desarrolló la prueba, su nombre, número de cédula de identidad y cantidad de hojas -en ese orden-; las demás hojas es suficiente con nombre, número de cédula y número de página.

Problema 1	19 (7, 8, 4) ptos	

Para resolver un problema particular (utilizando *Matlab*) se necesita trabajar con una estructura que permita almacenar vectores de distinto largo. Se optó por utilizar una **matriz de vectores de largo variable (MVLV)**, de dimensiones **N**x**M**, donde cada fila está estructurada de la siguiente manera:

Largo del vector	Vector	Ceros

N (cantidad de filas) es igual a la cantidad de vectores en la colección. **M** (cantidad de columnas) es igual al largo del vector con más elementos de la colección + 1.
Por ejemplo, la colección de vectores [2,3], [1,0,41], [5,13], [9,1,0,0,6,1], [1,0] se guardaría de la siguiente manera:

2	2	3	0	0	0	0
3	1	0	41	0	0	0
2	5	13	0	0	0	0
6	9	1	0	0	6	1
2	1	0	0	0	0	0

Se pide:

a) Implementar en *Matlab* la función **iterativa** *D = insvec(C, v)* que, dada una MVLV *C* y un vector *v*, devuelve otra MVLV *D* producto de la inserción de *v* al final de *C*. Ejemplo de la invocación sucesiva de *insvec* en la Ventana de Comandos de *Matlab*:

Comandos	Resultado
`>> D = insvec([], [2,3])`	`D = [2,2,3]`
`>> D = insvec(D, [1,0,41])`	`D = [2,2,3,0;3,1,0,41]`
`>> D = insvec(D, [5,13])`	`D = [2,2,3,0;3,1,0,41;2,5,13,0]`

b) Implementar en *Matlab* la función **recursiva** *v = darveclargo(C, largo)* que retorna el primer vector almacenado que tenga el largo especificado. Si no se encuentra el vector, la función debe retornar []. Ejemplo:

```
>> v = darveclargo(D, 3)
                    v = [1,0,41]
```

c) Implementar en *Matlab* una función **iterativa** *[Di, Dj, Ds] = insvecdisp(Ci, Cj, Cs, v)* que resuelve el mismo problema que *insvec* pero que trabaja con MVLV dispersas (por ejemplo, la MVLV dispersa *C* está representada en formato elemental por los vectores *Ci*, *Cj*, *Cs*).

Nota 1: Recordar que los elementos iguales a cero del vector *v* (en caso de que existan) no deben almacenarse en la matriz dispersa.
Nota 2: Se puede asumir que la matriz dispersa está ordenada por el vector de filas (*Ci*) de menor a mayor.
Nota 3: No se puede usar **`sort`**, **`find`**, **`sparse`**, ni ninguna otra función que resuelva trivialmente el problema.
Nota 4: Se puede usar operadores que generen matrices elementales.

Problema 2	12 ptos	

Se pretende optimizar el sistema de cajas registradoras de un supermercado agregándole una funcionalidad que le permita al cajero saber rápidamente la forma óptima de dar un vuelto. Para ello se necesita una función **recursiva** tal que, dados una cifra (que representa el importe a devolver) y un vector de valores de billetes y monedas, devuelva un vector que indique la menor cantidad de billetes y/o monedas de cada valor que se necesitan para formar la cifra especificada.

Ejemplo:
PARAMETROS DE ENTRADA:
- Cifra: 1753.50
- Valores: [1000, 500, 200, 100, 50, 20, 10, 5, 1, 0.50, 0.10]

SALIDA ESPERADA:
- Cantidades por billete: [1, 1, 1, 0, 1, 0, 0, 0, 3, 1, 0]

Nota 1: El vector de valores de billetes y monedas siempre viene ordenado de mayor a menor, como en el ejemplo.
Nota 2: Se puede asumir que se dispone de la cantidad suficiente de cualquiera de los billetes y monedas.

Problema 3	13 (3,5,5) ptos	

Los polinomios de Lucas se definen recursivamente de la siguiente manera:

$L_0(x) = 2$

$L_1(x) = x$

$L_N = x.L_{N-1}(x) + L_{N-2}(x)$

De desea implementar una función en *Matlab* que retorne el N-ésimo polinomio de Lucas. Para ello se pide realizar las siguientes funciones:

a) Implemente una función en *Matlab* que tome como entrada un polinomio *p(x)* en formato *Matlab* y devuelva un polinomio *q(x)* en formato *Matlab* tales que $q(x) = x.p(x)$.

b) Implemente una función **iterativa** en *Matlab* que tome como entrada dos polinomios p(x) y q(x) en formato *Matlab* y devuelva la suma de ambos en formato *Matlab*. Recuerde eliminar los elementos de más a la izquierda cuyo coeficiente valga 0.

c) Implemente una función **recursiva** en *Matlab* que tome como entrada un natural N y devuelva el N-ésimo polinomio de Lucas.

Nota 1: Para la resolución de este ejercicio no se permite el uso de ninguna función *Matlab* que resuelva de manera trivial el problema.
Nota 2: En la resolución de la parte c, se recomienda invocar las funciones de las partes a y b.