COMPUTACIÓN 1
Instituto de Computación

2do Parcial - 28 de noviembre de 2019

- Duración del parcial: 3:00 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 parcial.
- Escriba las hojas de un solo lado. Las partes no legibles del examen se considerarán no escritas.
- En la primera hoja a entregar ponga con letra clara, en el ángulo superior derecho, su **nombre**, número de **cédula de identidad** y **cantidad de hojas** -en ese orden-; en las demás hojas pongan nombre, número de cédula y número de página.

Para la resolución de los diferentes ejercicios **solamente** podrá utilizar las siguientes funciones brindadas por **Octave**:

- `length() y size()`
- `mod() y rem()`
- `floor(), ceil() y round()`
- `zeros() y ones()`

Nota: En todos los ejercicios se deben usar las estructuras de control adecuadas para cada caso.

Problema 1	8 (1, 1, 2, 2, 2) ptos	

a) Represente el número -15 en complemento a 1 con 5 bits.

b) Realice la suma de los siguientes enteros en complemento a 2 con 5 bits: 10011 y 10010.

c) En un sistema de punto flotante con 1 bit de signo, 3 bits de exponente y 5 de mantisa, donde el exponente 000 se reserva para los números desnormalizados y el 111 se reserva para Infinito y NaN:

I. ¿Cuántos números en punto flotante (normalizados) son representables entre el 4 y el 8 (incluyendo el 4 y el 8)? Justifique

II. ¿Cuál es el decimal representado por la tira 1 101 10000?

III. Restar los siguientes números representados en el sistema anterior:

0 110 11101

1 100 10000

Problema 2	12 (4,8) ptos	

a) Implementar en *Octave* la función ***recursiva*** *Suma* que, dado un vector retorna la suma de sus elementos.

b) Implementar en *Octave* la función ***recursiva*** *SumaParImpar* que, dado un vector de enteros, retorne dos números: la suma de los elementos en posiciones impares y la suma de los elementos en posiciones pares.

```
SumaParImpar([1,6,2,5,4]) = [1+2+4,6+5]=[7,11]
SumaParImpar([1]) = [1,0]
SumaParImpar([]) = [0,0]
```

Problema 3	8 ptos	

Implemente la función ***iterativa*** *esSimetrica* que reciba una matriz cuadrada y devuelva 1 si la matriz es simétrica y 0 sino. La función debe recorrer la menor cantidad posible de elementos.

Problema 4	8 ptos	

Decimos que una posición i de un vector es la "reflexión horizontal" de otra posición k si se cumple que:

```
i == length(v) - k + 1
```

Es decir, la primer posición es la reflexión horizontal de la última, la segunda de la penúltima, etc.

Implementar en *Octave* la función ***recursiva*** *sumaReflexiones_rec*, que dado un vector *v*, retorne un vector *u* tal que para todo i ≤ k donde i es la reflexión horizontal de k en *v, u(i)=v(i)+v(k)*.

Ejemplos:

```
sumaReflexiones([1,6,2,5]) = [1+5,6+2]=[6,8]
sumaReflexiones([1,6,3,2,5]) = [1+5,6+2,3+3]=[6,8,6]
sumaReflexiones([]) = []
```

Problema 5	8 ptos	

Implementar en *Octave* la función ***iterativa*** *Combinaciones* que, dado un natural, *k*, devuelva una matriz con una columna por cada combinación posible de largo *k* de 0's y 1's.

Ejemplo:

Las combinaciones de largo 3 de 0's y 1's es una matriz de dimensión 3 x 2^3:

0	0	0	0	1	1	1	1
0	0	1	1	0	0	1	1
0	1	0	1	0	1	0	1

Consejo: observe el patrón de 0's y 1's de la matriz en el ejemplo. Puede utilizar las funciones de Octave *zeros(m,n)* y *ones(m,n)*.

Problema 6	16 (8,8) ptos	

a) Para una matriz dispersa *A*, y un vector denso *x*, la operación *y = A*x* (producto matriz-vector) puede calcularse de la siguiente manera:
paso 1) inicializar *y* como un vector de ceros
*paso 2) para todo i,j tal que A(i,j) es distinto de 0, y(i) = y(i) + A(i,j)*x(j)*

Implementar en *Octave* la función ***iterativa*** *spmv_it* donde a partir de una matriz dispersa de tamaño *m* x *n* **en formato elemental** y un vector de tamaño *n*, devuelva un vector de tamaño *m* que sea el resultado de la multiplicación de la matriz por el vector de entrada. La función debe recibir por parámetro la matriz en formato elemental, el vector y *m*.

b) Implementar en *Octave* la función ***recursiva*** *mult_col_rec* que reciba una matriz dispersa de tamaño *m* x *n* **en formato elemental** y un vector *v* de tamaño *n*, devolviendo una matriz dispersa **en formato elemental** con el resultado de multiplicar cada elemento de la columna *i* de la matriz por *v(i)*, para cada elemento *i* del vector *v*. Notar que *v* puede contener 0's.