2do Parcial - 29 de noviembre de 2018

- 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()`

Problema 1	8 (1, 2, 2, 3) ptos	

a) Represente el número -16 en complemento a 1 con 5 bits.

b) Determine el número en base 10 que representa la tira 0 10000011 00000000000000000000000 en punto flotante de simple precisión de IEEE.

c) Escriba el número 65 en el sistema de punto flotante de simple precisión de IEEE.

d) Sumar los siguientes números representados en punto flotante de simple precisión de IEEE:

1 10000010 00011000000000000000000

1 10000001 10000000000000000000000

Problema 2	10 (6, 4) ptos	

Considere el código de la función recursiva *Rara*:

```
function res = Rara(n)
  if n == 0
     res = 1;
  elseif n == 1
     res = 2;
  else
     res = Rara(n-2)*Rara(n-1) + n;
  end
endfunction
```

Asumiendo que el orden de evaluación de las expresiones es de izquierda a derecha, responda las siguientes preguntas:

a) Escriba en orden todas las invocaciones que se hace a la función *Rara* cuando se invoca desde la consola `Rara(4)`. ¿Cuál es el resultado de invocar `Rara(4)`?

c) Implementar en *Octave* la función ***recursiva*** **cuadradoRec** que dado un número natural *n*, devuelva el cuadrado de *n*. Tenga en cuenta que para cualquier número natural *n* se cumple que $(n - 1)^2 = n^2 - 2n + 1$, por lo que despejando es posible calcular $n^2 = (n - 1)^2 + 2n - 1$.

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

Problema 3	6 ptos	

Implementar en *Octave* la función *iterativa* **sumaPolIt** que dados dos polinomios *p* y *q* con todos sus coeficientes no negativos, devuelva un polinomio *s* que sea la suma de *p* y *q*. **Nota**: *p* y *q* no pueden ser vacíos y pueden ser de distinto grado.

Ejemplos:

```
>> sumaPolIt([1,2,3],[4,3,0,2,5])
ans = [4,3,1,4,8]
```

Problema 4	6 ptos	

Implementar en *Octave* la función *iterativa* **alMenosIt** que dado un vector *v*, un elemento *elem* y un entero no negativo *n*, devuelve 1 si *elem* pertence al vector *v* al menos *n* veces, y 0 en caso contrario. La función debe procesar solamente los elementos necesarios para devolver el resultado.

Ejemplos:

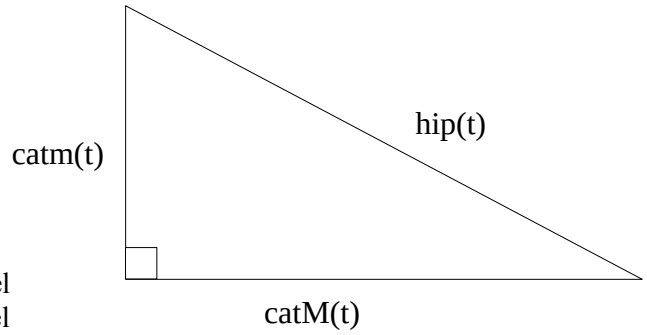
```
>> alMenosIt([],4,0)
ans = 1

>> alMenosIt([],4,2)
ans = 0

>> alMenosIt([1,4,2,4,3],4,0)
ans = 1

>> alMenosIt([1,4,2,4,3],4,1)
ans = 1

>> alMenosIt([1,4,2,4,3],4,3)
ans = 0
```

Problema 5	8 ptos	

Implementar en *Octave* la función *iterativa* **ladosTrianguloIt** que reciba un entero positivo *n* y devuelva tres vectores catm, catM e hip con todos los triángulos rectángulos "t" cuyos catetos catm(t) y catM(t), y cuya hipotenusa es hip(t), cumplen que son enteros menores o iguales a *n*. De esta forma, para cada posición "t" de los vectores resultado se debe cumplir que catm(t)^2 + catM(t)^2 = hip(t)^2 y también se cumple que $1 \leq catm(t) < catM(t) < hip(t) \leq n$.

Ejemplos:

```
>> [catm, catM, hip]=triang(25)
catm = [3,5 ,6 ,7 ,8 ,9 ,12,15]
catM = [4,12,8 ,24,15,12,16,20]
hip  = [5,13,10,25,17,15,20,25]
```



Es decir que se pueden construir siete triángulos, el primer triángulo tiene catetos 3 y 4 e hipotenusa 5, el segundo triángulo tiene catetos 5 y 12 e hipotenusa 13, etc.

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

Problema 6	8 ptos	

La sucesión de Fibonacci es:
fib(1) = 1
fib(2) = 1
fib(n) = fib(n-1) + fib(n-2), para todo n > 2

Implementar en *Octave* la función *recursiva* **fibVectRec** que dado un número entero *n* mayor que cero, devuelva un vector con los *n* primeros elementos de la sucesión de Fibonacci. La implementación de esta función no puede usar ni *for* ni *while.*

```
>> fibVectRec(1)
ans = [1]
>> fibVectRec(2)
ans = [1, 1]
>> fibVectRec(3)
ans = [1, 1, 2]
>> fibVectRec(4)
ans = [1, 1, 2, 3]
```

Problema 7	14 (6,8) ptos	

a) Implementar en *Octave* la función *recursiva* **suma_por_columna_rec** que dada una matriz dispersa en formato elemental y un entero *n* con la cantidad de columnas de la matriz, devuelva un vector de largo *n* con la suma por columnas de la matriz.

Ejemplos (en los ejemplos el orden de los vectores es fila, columna, valor):

```
>> suma_por_columna_rec([],[],[],4)
ans = [0, 0, 0, 0]
>> suma_por_columna_rec([6,9,25],[3,1,3],[4.1,2.3,1.7],4)
ans = [2.3, 0, 5.8, 0]
>> suma_por_columna_rec([6,9,25,25,54],[3,1,3,5,2],[4.1,2.3,1.7,3.2,9.1],8)
ans = [2.3, 9.1, 5.8, 0, 3.2, 0, 0, 0]
```

b) Implementar en *Octave* la función *recursiva* **seleccionar_columnas_rec** que dada una matriz dispersa en formato elemental y un entero *col*, devuelva una matriz dispersa que contenga solamente los elementos de la columna *col.*

Ejemplos (en los ejemplos el orden de los vectores es fila, columna, valor):

```
>> [f,c,d] = seleccionar_columnas_rec([],[],[],4)
f = []
c = []
d = []
>> [f,c,d] = seleccionar_columnas_rec([6,9,25,25],[3,1,3,5],[4.3,2.7,1.5,3.0],8)
f = []
c = []
d = []
>> [f,c,d] = seleccionar_columnas_rec([6,9,25,25],[3,1,3,5],[4.3,2.7,1.5,3.0],5)
f = [25]
c = [5]
d = [3.0]
>> [f,c,d] = seleccionar_columnas_rec([6,9,25,25],[3,1,3,5],[4.3,2.7,1.5,3.0],3)
f = [6,25]
c = [3,3]
d = [4.3,1.5]
```