COMPUTACIÓN 1
Instituto de Computación
Examen - 17 de diciembre de 2021

- Duración del examen: 2 hs 30 minutos.
- No se podrá utilizar ningún tipo de material (apuntes, libro, calculadora, etc). Apague su celular.
- **Sólo** se contestarán preguntas sobre interpretación de la letra hasta 30 minutos antes de la finalización del mismo.
- Las partes no legibles del parcial se considerarán no escritas
- En la primer 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-; las demás hojas es suficiente con 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()
- abs()
- zeros() y ones()

Notas: - No se deben realizar más iteraciones ni invocaciones recursivas que las necesarias para resolver el problema
- En todos los ejercicios se deben usar las estructuras de control adecuadas para cada caso.
Por ejemplo: se controlará el uso correcto de for y while.

Problema 1	16 ptos (4,4,4,4)	

a) Calcule la suma de los siguientes números representados en exceso a $M=2^5$ con 6 bits: 010010+011000

b) Convierta el número -65 a Complemento a 2 de 8 bits.

c) Complete la siguiente función que suma 1 al número binario representado por el vector v, devolviendo también el acarreo:

```
function [v, acarreo] = sumar_1(v)

  acarreo = 1;
  n = length(v);
  while n>=1 && acarreo
    bit= % completar ...
    acarreo= % completar ...
    v(n)=bit;
    n=n-1;
  endwhile
```

d) En punto flotante de 1 bit de signo 4 bits de exponente y 7 bits de mantisa, donde el exponente se expresa en exceso M=7, 0000 se reserva para números desnormalizados y 1111 para infinito y NaN, sumar los siguientes números y expresar el resultado en decimal: 010010010000+110001000000

En todos los casos debe justificar su respuesta

Problema 2	14 ptos	

Escribir en Octave una función **recursiva** `esRaiz` que dado un polinomio ***pol*** en formato Octave y un número ***x***, devuelva 1 si x es raíz de pol y 0 en caso contrario.

Se sugiere resolver el problema utilizando una función cabezal.

COMPUTACIÓN 1
Instituto de Computación

Problema 3	17 ptos	

Escriba en Octave la función **iterativa** buscarDosQueSumenS, que dado un vector ***v***, y un número ***s***, devuelva dos enteros ***i*** y ***j*** tal que v(i)+v(j)=s. Si no existen dos elementos de v que cumplan con la condición, la función devuelve -1 en ambos parámetros de salida. ***i*** puede ser igual a ***j.*** Ejemplos:

```
>> [i,j] = buscarDosQueSumenS([1, 5, 2, 6, -3, 10], 3)
>> i = 1
>> j = 3
>> [i,j] = buscarDosQueSumenS([1, 5, 2, 6, -3, 10], 35)
>> i = -1
>> j = -1
```

Problema 4	37 ptos (12,8,17)	

a) Escriba en Octave la función **recursiva** posMin_rec que dado un vector de números ***v*** con por lo menos un elemento devuelve el mínimo del vector y su posición. Si el elemento mínimo se encuentra más de una vez se puede devolver cualquiera de sus posiciones.

b) Escriba en Octave la función **iterativa** reemplazarMin que dado un vector de números ***v*** con por lo menos un elemento y un número ***x,*** reemplaza el mínimo ***m*** de ***v*** por ***x*** si y solo si ***x*** > ***m.*** Puede asumir que ***v*** no tiene elementos repetidos. Ejemplos:

```
>> v = reemplazarMin([1, -5.5, 2, 6.3, 3, 10], 0)
  v = [1, 0, 2, 6.3, 3, 10]
>> v = reemplazarMin([1, -5.5, 2, 6.3, 3, 10], -42)
  v = [1, -5.5, 2, 6.3, 3, 10]
```

c) Escriba en Octave la función **iterativa** nMayores que dado un entero ***n*** y un vector de números ***v*** no vacío y con al menos ***n*** elementos, devuelve un vector con los ***n*** mayores elementos de ***v***. Para esta función se sugiere utilizar reemplazarMin aunque no la haya resuelto. El vector que se devuelve puede tener cualquier orden. Ejemplo:

```
>> v = nMayores([1, -5.5, 2, 6.3, 3, 10], 3)
  v = [6.3,3,10]
```

Problema 5	16 ptos	

Representaremos un laberinto como una matriz donde hay un 0 en los lugares por donde es posible pasar y un 1 por donde no. En cada posición del laberinto se puede avanzar un paso en 4 direcciones (arriba, abajo, izquierda y derecha). Escriba una función **recursiva** haySalida en Octave que dado un laberinto, una posición inicial representada por dos enteros ***i_0*** y ***j_0***, y una posición final representada por ***i_f*** y ***j_f***, devuelva 1 si es posible llegar desde la posición inicial a la final, devolviendo 0 en caso contrario. Ejemplo:

```
laberinto = [0 1 1 1 0 0 1 0;
             0 0 0 1 1 0 1 0;
             1 1 0 0 1 1 1 1;
             0 1 1 0 0 0 1 1;
             0 1 0 0 1 0 0 0]
haySalida(laberinto,1,1,5,8) = 1 % la posición inicial es (1,1)
haySalida(laberinto,4,1,5,8) = 0 % la posición inicial es (4,1)
```