COMPUTACIÓN 1
Instituto de Computación

Segundo Parcial – 29 de noviembre de 2024

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.
Escriba las hojas de los dos lados. 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 las demás hojas pongan nombre, número de cédula y número de página.

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

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

Notas: : - 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.
- No se deben realizar más iteraciones o invocaciones recursivas que las necesarias para resolver el problema

Problema 1	11 (2, 2, 2, 2, 3) ptos	

En todos las siguientes partes deben justificarse los resultados:

a) Escriba el número en base 2 correspondiente al decimal $d = (2^4\text{-}1) \times 2^5 + 12$

b) Escriba el número en base 16 resultado de restar $2FAB34_{16}\text{-}1E9A23_{16}$

c) Escriba el resultado de 1000100 - 0001001 trabajando en complemento a 1 de 7 bits

d) Escriba el resultado de -3,25 x 2^4 en complemento a 2 de 7 bits

e) Escriba el resultado de la operación 000100000000 x 110010000000 trabajando en punto flotante con 1 bit de signo, 4 bits de exponente almacenado en exceso a M = 7

Problema 2	5 ptos	

Considere la siguiente función:

```
function res = func(lista, n)
  largo = length(lista);
  if largo==0
    res = [];
  elseif mod(lista(1),2)==0
      lista = lista(2:largo);
      n = n - 1;
      res = [n, func(lista,n)];
  else
      x = lista(1);
      lista = lista(2:largo);
      res = [func(lista,n-1),x];
  end
end
```

¿Qué valor queda almacenado en `z, res` y `lista` como resultado de las siguientes invocaciones?:

```
lista = [4,2,3,5];
n = 3;
z = func(lista,n);
```

Problema 3	10 ptos	

Implementar en *Octave* la función ***recursiva*** *cambiarUnosCeros* que, dado un vector *v* que contiene números enteros, devuelva un vector del mismo largo sustituyendo los ceros por unos y los unos por ceros, sin cambiar los demás valores.

Ejemplos de ejecución:

```
cambiarUnosCeros([]) → []
cambiarUnosCeros([0,1,3,1,0,2,1]) → [1,0,3,0,1,2,0]
cambiarUnosCeros([1,0,1]) → [0,1,0]
cambiarUnosCeros([2,8]) → [2,8]
```

Problema 4	12 (6, 6) ptos	

a) Implementar en *Octave* la función ***iterativa*** *trazaCompleta* que dada una matriz cuadrada completa *m* retorne la traza de dicha matriz. Recordar que la traza de una matriz cuadrada está definida como la suma de los elementos de la diagonal principal de la matriz.

b) Implementar en *Octave* la función ***iterativa*** *trazaDispersa* que dada una matriz cuadrada dispersa en formato elemental (Md, Mf, Mc) retorne la traza de dicha matriz. Recordar que la traza de una matriz cuadrada está definida como la suma de los elementos de la diagonal principal de la matriz.

Problema 5	10 ptos	

Implementar en *Octave* la función ***recursiva*** *primeroMayorQueX* que, dado un vector *v* de números enteros y positivos, y un número *x*, devuelva el primer elemento del vector que es mayor que *x*. En caso de no existir ningún elemento mayor que *x* en el vector, la función deberá devolver -1.

Ejemplos de ejecución:

```
primeroMayorQueX([],-5) → -1
primeroMayorQueX ([6,10,24,1],-3) → 6
primeroMayorQueX ([6,10,24,1],10) → 24
primeroMayorQueX ([6,10,24,1],30) → -1
```

Problema 6	12 ptos	

Implementar en Octave la función **recursiva** *posSecUnosCeros* que, dado un vector *v* cuyas entradas contienen únicamente 1's y 0's, devuelva dos vectores *posUnos* y *posCeros* con la posición inicial de cada secuencia de 1's y 0's, respectivamente. Si no hay ninguna secuencia de 1's, o 0's, el vector correspondiente es vacío.

Ejemplos de ejecución:

```
posSecUnosCeros([1,1,1,0,0,1,1,1,1,0,1,1,1,1,0,0,0,1,0,1]) →
posUnos=[1,6,11,18,20], posCeros=[4,10,15,19]

posSecUnosCeros([0,0,0,0,0,0,0]) → posUnos=[], posCeros=[1]

posSecUnosCeros([1,1,1,1,1,1,1]) → posUnos=[1], posCeros=[]

posSecUnosCeros([]) → posUnos=[], posCeros=[]
```