COMPUTACIÓN 1
Instituto de Computación

Segundo Parcial – 29 de noviembre de 2022

- 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 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 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()`
- `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	10 (3, 1, 1, 2, 3) ptos	

En todos los siguientes ejercicios deben justificarse los resultados:

a) Determine *a*, *b* y *c* para que $2^a+2^b-2^c$ sea igual al valor decimal de 1000011111111_2.

b) Calcule el resultado de 10000010 - 10000001 en Comp a 1 de 8 bits (indicando si ocurre overflow).

c) Calcule el resultado de 11110001 - 11110010 en Comp a 2 de 8 bits (indicando si ocurre overflow).

d) Represente el número 1010011_2 en sistema cuaternario (base 4).

e) Convierta el decimal 4,25 a punto flotante con 1 bit de signo, 4 bits de exponente y 7 bits de mantisa. El exponente se representa con desplazamiento M=7.

Problema 2	5 ptos	

Considere la siguiente función:

```
function f = func(n,lista)
  if n==length(lista)
    f=lista(n);
  else
    lista(n+1)=lista(n+1)+lista(n);
    f=func(n+1,lista);
  end
endfunction
```

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

```
lista = [1,2,3,4,5];
f=3;
z = func(f,lista);
f=f+z;
```

Problema 3	8 ptos	

Implementar en *Octave* la función ***recursiva*** *maxBilat* que dado un vector ***v*** devuelva el máximo de todas las sumas ***v(i)+v(n-i+1)*** (***n*** es el tamaño de ***v***). El vector ***v*** tiene al menos un elemento.

Ejemplo:
`v=[1,5,30,4,-5,6,7]`
`maxBilat(v)` compara 1+7,5+6,30-5, 4+4, y devuelve 25, que es el valor máximo de todas las sumas.

Problema 4	10 ptos	

La matriz M representa un campo minado, donde M(i,j)=1 si la coordenada contiene una mina y 0 en caso contrario. Se desea saber, dada una lista de casillas de la matriz, si es posible recorrerlas para poner minas en dichas casillas, es decir, ***las casillas de la lista no deben contener minas y no se pueden volver a pisar una vez que ya se visitaron***.
Implementar en Octave la función **recursiva** *puedoPonerMinas* que dada una matriz de dos columnas que representa una lista de casillas y un campo minado *M*, devuelva un 1 si es posible recorrer la lista de coordenadas poniendo minas y un 0 en caso contrario.

Ejemplos:

```
M = [1 0 0 0;
     1 0 1 0;
     0 0 0 0;
     1 0 1 1]

puedoPonerMinas([2,2;2,1;3,3],M) → 0 (ya que M(2,1) contiene una mina)
puedoPonerMinas([1,2;2,2;3,3],M) → 1
puedoPonerMinas([1,2;2,2;3,3;2,2],M) → 0 (ya que vuelve a pasar por M(2,2))
```

Problema 5	(5,10) ptos	

a) Escriba en Octave una función ***iterativa*** sumoUnoIt(v) que le sume 1 a cualquier binario no negativo representado como un vector ***v*** de 0s y 1s. El vector ***v*** tiene al menos un elemento.

b) Escriba el equivalente ***recursivo*** sumoUnoRec(v) de la parte a).

Nota: Sumar 1 a un binario no negativo equivale a invertir todos los dígitos desde el menos significativo hasta el primer 0 inclusive. Si no hay ningún cero se sustituyen todos los dígitos por 0 y se agrega un 1 a la izquierda.

Ejemplos:

```
sumoUnoIt([1 0 0]) -> [1 0 1]
sumoUnoIt([1 1]) -> [1 0 0]
sumoUnoIt([0 1]) -> [1 0]
sumoUnoIt([0 0 0]) -> [0 0 1]
sumoUnoIt([1]) -> [1 0]
sumoUnoIt([0]) -> [1]
```

Problema 6	12 (6,6) ptos	

En una plataforma de películas se representa qué películas vio cada usuario mediante tres vectores *usuario*, *peli*, y *veces*, donde si un usuario ***u*** vio al menos una vez una película ***p***, existe un índice ***i*** para el que *usuario(i)=u*, *peli(i)=p* y *veces(i)=v* donde ***v*** es la cantidad de veces que *u* vio *p*.

Ejemplo:
```
usuario=[1 2 1 3 2]
peli   =[1 3 4 2 4]
veces  =[1 2 1 1 1]
```
El usuario 1 vio las películas 1 y 4 (ambas 1 vez). El usuario 2 vio la película 3 (2 veces) y la película 4 (1 vez). El usuario 3 vio la película 2 (1 vez).

a) Escriba en Octave una función ***recursiva*** *contarRepro* que reciba el vector *peli*, el vector *veces* y la cantidad total de películas *cantP*; y devuelva un vector de tamaño *cantP* que en la posición ***p*** contenga la cantidad de reproducciones de la película ***p***.
Nota: para resolver esta parte puede ser útil tener en cuenta que en caso de que ningún usuario haya visto películas en la plataforma, la solución es un vector de ceros de tamaño cantP.

b) Escriba en Octave una función ***recursiva*** *darDeBaja* que reciba los vectores *usuario*, *peli*, *veces* y un usuario *u;* devolviendo los vectores *usuario*, *peli* y *veces* sin las entradas correspondientes al usuario *u.*