COMPUTACIÓN 1
Instituto de Computación

Primer Parcial – 19 de setiembre de 2023

- Duración del parcial: 2:30 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 AMBOS 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 diferentes ejercicios **solamente** podrá utilizar las siguientes funciones brindadas por **Octave**:

- `length() y size()`
- `mod(), div() y rem()`
- `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 que las necesarias para resolver los problemas.

Problema 1	4 (2, 2) ptos	

En este ejercicio `x` y `y` son variables escalares (no vectores) booleanas (sus valores son 1 o 0).

a) Reescribir la siguiente expresión lógica no utilizando el operador `|` ni `||`:

```
r = ~(x || y)
```

b) Reescribir la siguiente expresión lógica no utilizando los operadores &&, &, `||` ni `|`:

```
r = x && (x || y)
```

Problema 2	4 ptos	

Determine el valor de las variables `a,b` y `c` luego de ejecutar `miscript.m` desde la línea de comandos de Octave.

```
% func.m
function c = func(a)
  c = 0;
  for i = 1:a
    c = c + i;
  end
end
```

```
% miscript.m
b = 5;
a = b;
a = func(a);
```

Problema 3	8 ptos	

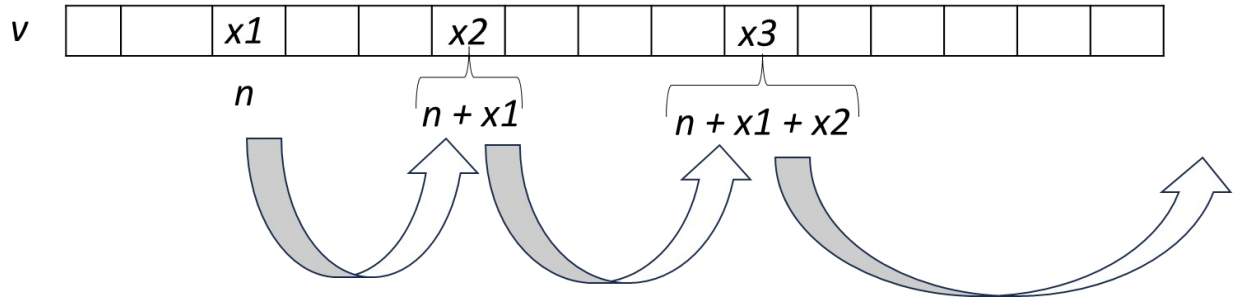
Implementar en Octave la función *estadisticasCurso* que, dado un vector *v1* que contiene las calificaciones correspondientes a un curso en la escala del 1 al 100 de cada uno de los estudiantes, devuelve un vector *v2* de largo 3, donde en la primera posición aparece la cantidad de estudiantes que deben recursar (sacaron menos de 25), en la segunda posición aparece la cantidad de estudiantes habilitados a dar el examen (sacaron entre 25 y 59) y en la tercera posición aparece la cantidad de estudiantes que exoneraron (sacaron más de 59).

Ejemplos de ejecución:

```
estadisticasCurso([52,20,60,25,81,71]) → [1,2,3]
estadisticasCurso([59,60,25,81,71,60,55,51]) → [0,4,4]
```

Problema 4	8 ptos	

Implementar en *Octave* la función *contarElementosASaltos* que, dado un número *n* entero y positivo y un vector *v* de números enteros positivos, devuelva la cantidad de elementos que se pueden visitar sin salirse del vector del siguiente modo: la primera posición que se visita es *n*, siempre y cuando caiga dentro del vector, y se obtiene el valor *x1*, la segunda posición que se visita, siempre y cuando caiga dentro del vector, es *n+x1* y se obtiene el valor *x2*, la tercera posición que se visita, siempre y cuando caiga dentro del vector, es *n+x1+x2* y así sucesivamente.



Ejemplos de ejecución:

```
contarSaltos([1,1,1,1,1,1],7) → 0
contarSaltos([1,1,1,1,1,1],6) → 1
contarSaltos([1,1,1,1,1,1],1) → 6
contarSaltos([2,1,2,1,2,1],1) → 3
contarSaltos([1,2,3,4,5],2) → 2
contarSaltos([],2) → 0
```

Problema 5	8 ptos	

Implementar en *Octave* la función *mayorMenor* que, dados un vector de números enteros y positivos **ordenado de forma ascendente** *v,* y un número *n* entero y positivo, devuelva el mayor valor del vector que es menor que *n.* Si *n* es menor que todos los elementos del vector se debe devolver -1.

```
mayorMenor([1,2,20,34,41], 33) → 20
mayorMenor([1,2,20,34,41], 34) → 20
mayorMenor([1,2,20,34,41], 56) → 41
mayorMenor([5,13,17], 4) → -1
mayorMenor([5,13,17], 9) → 5
```

Problema 6	8 ptos	

Implementar en Octave la función *mqSuma* que, dado un vector *pos* de números positivos y negativos, devuelva un vector *indices* con todas las posiciones del vector *pos* cuyo valor es mayor que la suma de los siguientes números en *pos*. Es decir, si $pos(i) > (\sum_{j=i+1}^{n} pos(j))$, siendo *n* el largo del vector *pos*, entonces el índice *i* tiene que aparecer en el vector *indices.* Nota: el índice *n* nunca aparece en el vector *indices.* Si ninguna posición del vector *pos* cumple con la condición se debe devolver el vector vacío.

```
mqSuma([2,4,5,1]) → [3]
mqSuma([2,4,-5,1]) → [1,2]
mqSuma([2,4,-5,-6]) → [1,2,3]
mqSuma([-2,4,-5,6,-3,-2,5]) → [2,4]
mqSuma([5 5 5]) → []
```