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)

010010

+011000

101010

a esto debemos restarle el exceso:

101010

-100000

001010

b)

65: 01000001

-65: 10111110+1= 10111111

d)

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

  acarreo = 1;
  n = length(v);
  while n>=1 && acarreo
     bit=mod(v(n)+1,2); %también puede ser (v(n)==0)
     acarreo=(v(n)+1==2); %también puede ser (~bit) o (v(n)==1)
     v(n)=bit;
     n=n-1;
  endwhile
```

e)

100010000000 es negativo y es el menor en valor absoluto de los dos números, por lo que invertimos el signo y se lo restamos al primero. El signo del resultado será positivo. El primer paso es igualar el exponente del segundo al del primero:

1001 1,0010000

- 1001 0,1100000

1001 0,0110000 — Normalizando: 0 0111 1000000 = $1,1_2 x 2^o = 1,5_{10}$

Problema 2	14 ptos	

```
%% esRaiz: cabezal de la funci'on
function [res] = esRaiz(pol,x)
        res = polEval_rec(pol,x)==0;
endfunction


function r = polEval_rec(pol,x)
        lp = length(pol);
        if (lp == 0)
                r = 0;
        else
                r = x*polEval_rec(pol(1:lp-1),x) + pol(lp);
        end
endfunction
```

Problema 3	17 ptos	

```
%% buscarDosQueSumenS: busca los primeros 2 elementos que suman s
function [i,j] = buscarDosQueSumenS(v,s)
        i = -1;
        j = -1;
        encontre=0;

        lv=length(v);

        x=1;
        while x <= lv && ~encontre
                y=x;
                while y <= lv && ~encontre
                        if v(x)+v(y)==s
                                i=x;
                                j=y;
                                encontre=1;
                        end
                        y=y+1;
                end
                x=x+1;
        end
endfunction
```

COMPUTACIÓN 1
Instituto de Computación

Problema 4	37 ptos (12,8,17)	

a)

```
%% posMin_rec: devuelve el minimo de un vector y la posicion en la que se encuentra
function [min,pos] = posMin_rec(v)

	lv=length(v);

	if lv == 1
		min=v(1);
		pos=1;
	else
		[min,pos]=posMin_rec(v(1:lv-1));
		if v(lv) < min
			min=v(lv);
			pos=lv;
		end
	end
endfunction
```

b)

```
%% reemplazarMin: reemplaza el m'inimo del vector solo si es menor que el elemento que se recibe por parametro
function [v] = reemplazarMin(v,x)
	lv = length(v);

	min=v(1);
	pos=1;

	for i=2:lv
		if v(i) < min
			min = v(i);
			pos = i;
		end
	end

	if x > min
		v(pos) = x;
	end
endfunction
```

c)

```
%% nMayores: devuelve los n mayores elementos del vector
% se asume length(v) >= n > 0
function res = nMayores(v,n)
	res = v(1:n);
	lv=length(v);
	for i=n+1:lv
		res=reemplazarMin(res,v(i));
	end
endfunction
```

Problema 5	16 ptos	

```
%% haySalida: devuelve 1 si hay un camino desde la posicion inicial a la final del laberinto
function [res] = haySalida(laberinto,i_0,j_0,i_f,j_f)
	[li,lj]=size(laberinto);
	if i_0 > li || i_0 < 1 || j_0 > lj || j_0 < 1 || laberinto(i_0,j_0)==1
		res = 0;
	elseif i_0==i_f && j_0==j_f
		res = 1;
	else
		laberinto(i_0,j_0)=1;
		res = haySalida(laberinto,i_0,j_0+1,i_f,j_f) || ...
			haySalida(laberinto,i_0,j_0-1,i_f,j_f) || ...
			haySalida(laberinto,i_0+1,j_0,i_f,j_f) || ...
			haySalida(laberinto,i_0-1,j_0,i_f,j_f);
	end
endfunction
```