EJERCICIOS DE EXPRESIONES

Expresiones aritméticas básicas

1. Dadas las siguiente expresiónes en JavaScript, para cada una aplicando las reglas de orden de agrupación (precedencia y asociatividad) y orden de evaluación:
   - pon paréntesis de forma que, sin modificar su funcionalidad, haga claro en qué orden se evalúa.
   - indica el valor resultante de la expresión y de las variables, si las hay

a)
```
15 - 3 - 2
```

b)
```
12 / 6 / 2
```

c)
```
3 ** 2 ** 3
```

d)
```
15 - 3 ** 4 / 3
```

e)
```
10/-0
```

f)
```
-10/-0
```

g)
```
10 * 10 ** 5 % 20 + 10 % 5
```

h)
```
let a = 10;
let b =
   a++ % 10 ? "ola " + a : "hola " + a;
```

i)
```
let a = 9;
let b =
   a++ % 10 ? "ola " + a : "hola " + a;;
```

j)
```
let a = 4;
a-- <= 5 % a  ? ++a : a--;
```

k)
```
Infinity / Infinity
```

l)
```
0/-0
```

m)
```
1_234 + 1_000
```

n)
```
1_000.34 + 1_000.10
```

o)
```
0xFE + 1
```

p)
```
0B001 + 0B001
```

q)
```
0o333 + 0o001
```

Expresiones aritméticas

2. Dadas las siguiente expresiónes en JavaScript, para cada una aplicando las reglas de orden de agrupación (precedencia y asociatividad) y orden de evaluación:
   - pon paréntesis de forma que, sin modificar su funcionalidad, haga claro en qué orden se evalúa.
   - indica el valor resultante de la expresión y de las variables, si las hay

a)
```
let a = 1;
a++ + --a;
```

b)
```
let a = 1;
a++ + a--;
```

c)
```
let a = 1;
++a + a--;
```

d)
```
let a = 1;
a > 1 ? a < 1 ? a++ : a-- : --a ;
```

e)
```
let a = 1;
let b = 2;
a * b >= 2 ? (a += b--, ++b) : a /= --b;
```

f)
```
let a = 1;
for (let i = 1, k = 10; i < 100; i+=k) {
   a += i;
}
```

Operadores lógicos de shortcut

3. Dadas las siguiente expresiónes en JavaScript, para cada una aplicando las reglas de orden de agrupación (precedencia y asociatividad) y orden de evaluación:
   - pon paréntesis de forma que, sin modificar su funcionalidad, haga claro en qué orden se evalúa.
   - indica el valor resultante de la expresión y de las variables, si las hay

a) let a = 1;
true || ++a
 	
b) let a = 1;
false || ++a
 	
c) let a = 1;
++a || true;
 	
d) let a = 1;
false && ++a;
	e) let a = 0;
	let b = 1;
	a++ && b++
	
	f) let a = 1;
	let b = 0;
	--a || b++;
	
	g) let a = 1;
	--a || a++;
	
	h) let a = 1;
	--a && a++;
		i) let a = 1;
		a-- && a++
		
		j) !(a && b) === !a || !b
		
		k) !(a || b) === !a && !b
		
		(estas 2 ultimas son las leyes de De Morgan)

Operadores a nivel de bit

4. Dadas las siguiente expresiónes en JavaScript, para cada una aplicando las reglas de orden de agrupación (precedencia y asociatividad) y orden de evaluación:
   - pon paréntesis de forma que, sin modificar su funcionalidad, haga claro en qué orden se evalúa.
   - indica el valor resultante de la expresión y de las variables, si las hay

a) Number.MAX_SAFE_INTEGER.toString(16) b) Number.MAX_SAFE_INTEGER.toString(2) c) (343).toString(2) d) ~0b0000000 e) ~0b11111111 f) ~0xFFFFFFFF g) ~0x80000000 h) ~1 i) ~~1 j) ~(!0) k) ~(!1) l) 0b11001111 & 0b10010011 m) 0b11001111 | 0b10010011 n) 0b11001111 ^ 0b10010011	o) 65 & 32 p) 65 | 32 q) 65 ^ 32 r) 0b01010101 ^ 0b10101010 s) 0b00000001<<1 t) (0b00000001<<1)<<1 u) ((0b00000001<<1)<<1)<<1 v) 0b00000001<<3 w) 0x80000000>>4 x) 0x80000000>>>4 y) (0x80000000 + 0x0fffffff)|0	z) ~false aa) ~true bb) ~~false cc) ~~true dd) ~undefined ee) ~~undefined ff) !(~false) gg) !(~~false) hh) ~a === -a - 1 ii) ~~(-0); jj) -0<<2>>2

Forzar aritmética entera

5. Dadas las siguiente expresiónes en JavaScript, para cada una aplicando las reglas de orden de agrupación (precedencia y asociatividad) y orden de evaluación:
   - pon paréntesis de forma que, sin modificar su funcionalidad, haga claro en qué orden se evalúa.
   - indica el valor resultante de la expresión y de las variables, si las hay

Estas expresiones buscan realizar las operaciones con aritmética entera y no aritmética de coma flotante

a) (19 / 2) (19 / 2)|0 (19 / 2)>>>0	c) ((19 / 2) + (23 / 3)) % 5 d) (((-19 / 2)|0 + (23 / 3)|0) % 5)

b) `(-19 / 2)`
`(-19 / 2)|0`
`(-19 / 2)>>>0`
	e) `(( ((19 / 2)|0) + ((23 / 3)|0 )) % 5)|0`
	
	f) `(( (-19 / 2)>>>0 + (23 / 3)>>>0 ) % 5)>>>0`

Forzar aritmética entera

6. Modificar la siguiente expresión en lenguaje C (que utiliza aritmética entera, al ser **d**, **m**, y **y** número enteros) para que funcione en JavaScript:

```
(d += m < 3 ? y-- : y-2 , 23 * m / 9 + d + 4 + y / 4 - y / 100 + y / 400 ) % 7
```

Asignación

7. Dadas las siguiente expresiónes en JavaScript, para cada una aplicando las reglas de orden de agrupación (precedencia y asociatividad) y orden de evaluación:
   - pon paréntesis de forma que, sin modificar su funcionalidad, haga claro en qué orden se evalúa.
   - indica el valor resultante de la expresión y de las variables, si las hay

a) `let a = b = 10;`
 	
b) `let a = b = (213434).toString(16);`
 	
c) `let a = 1;`
`true && (a = 31)`
 	
d) `const z = y = x = Math.max(1,3);`
`y++;`
`x++;`
`z++;`
	e) `let w = 20;`
	`let a = false && w++;`
	
	f) `let w = 20;`
	`let a = (false && w++);`
	
	g) `let w = 20;`
	`let a = true && w++;`
	
	h) `let w = 20;`
	`let a = (true && w++);`