EJERCICIOS DE EXPRESIONES

Expresiones aritméticas básicas

1. Dadas las siguiente expresiónes en PHP, 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) $a1 = 15 - 3 - 2;

b) $a2 = 12 / 6 / 2;

c) $a3 = 3 ** 2 ** 3;

d) $a4 = 15 - 3 ** 4 / 3;

e) $a5 = 10/-0;

f) $a6 = -10 / -0;

g) $a7 = 10 * 10 ** 5 % 20 + 10 % 5;

h) $a8 =  10;
$b8 =  $a8++ % 10 ? "ola ".$a8 : "hola
   ".$a8;

i) $a9 = 9;
$b9 = $a9++ % 10 ? "ola ".$a9 : "hola
   ".$a9;

j) $a10 = 4;
$r10 = $a10-- <= 5 % $a10  ? ++$a10 : $a10-
   -;
```

```
k) $r11 = INF / INF;

l) $r12 = 0/-0;

m) $r13 = 1_234 + 1_000;

n) $r14 = 1_000.34 + 1_000.10;

o) $r15 = 0xFE + 1;

p) $r16 = 0b001 + 0b001;

q) $r17 = 0333 + 0001;
```

Expresiones aritméticas

2. Dadas las siguiente expresiónes en PHP, 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) $a1 = 1;
$r1 = $a1++ + --$a1;

b) $a2 = 1;
$r2 = $a2++ + $a2--;

c) $a3 = 1;
$r3 = ++$a3 + $a3--;
```

```
d) $a4 = 1;
$r4 = $a4 > 1 ? ( $a4 < 1 ? $a4++ : $a4--) : --$a4 ;

e) $a5 = 1;
$b5 = 2;
$r5 = $a5 * $b5 >= 2 ? $a5 += $b5-- : $a5 /= --$b5;

f) $a6 = 1;
  $k6 = 10;
for ($i = 1; $i < 100; $i += $k6) {
    $a6 += $i;
}
```

Operadores lógicos de shortcut

3. Dadas las siguiente expresiónes en PHP, 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

```php
a) $a1 = 1;
$r1 = true || ++$a1;

b) $a2 = 1;
$r2 = false || ++$a2;

c) $a3 = 1;
$r3 = ++$a3 || true;

d) $a4 = 1;
$r4 = false && ++$a4;

e) $a5 = 0;
$b5 = 1;
$r5 = $a5++ && $b5++;

f) $a6 = 1;
$b6 = 0;
$r6 = --$a6 || $b6++;

g) $a7 = 1;
$r7 = --$a7 || $a7++;

h) $a8 = 1;
$r8 = --$a8 && $a8++;

i) $a9 = 1;
$r9 = $a9-- && $a9++;
```

```php
// Leyes de De Morgan:
j) $a10 = true;
$b10 = true;
$r10a = !($a10 && $b10) === (!$a10 || !$b10);
$r10aa = !($a10 && $b10) === !$a10 || !$b10;

k) $a10 = true;
$b10 = false;
$r10b = !($a10 && $b10) === (!$a10 || !$b10);
$r10bb = !($a10 && $b10) === !$a10 || !$b10;

l) $a10 = false;
$b10 = true;
$r10c = !($a10 && $b10) === (!$a10 || !$b10);
$r10cc = !($a10 && $b10) === !$a10 || !$b10;

m) $a10 = false;
$b10 = false;
$r10d = !($a10 && $b10) === (!$a10 || !$b10);
$r10dd = !($a10 && $b10) === !$a10 || !$b10;

n) $a11 = true;
$b11 = true;
$r11a = !($a11 || $b11) === (!$a11 && !$b11);
$r11aa = !($a11 || $b11) === !$a11 && !$b11;

o) $a11 = true;
$b11 = false;
$r11b = !($a11 || $b11) === (!$a11 && !$b11);
$r11bb = !($a11 || $b11) === !$a11 && !$b11;

p) $a11 = false;
$b11 = true;
$r11c = !($a11 || $b11) === (!$a11 && !$b11);
$r11cc = !($a11 || $b11) === !$a11 && !$b11;

q) $a11 = false;
$b11 = false;
$r11d = !($a11 || $b11) === (!$a11 && !$b11);
$r11dd = !($a11 || $b11) === !$a11 && !$b11;
```

Operadores a nivel de bit

4. Dadas las siguiente expresiónes en PHP, 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

```php
a) $r = 65 | 33;

b) $r = 65 ^ 3;

c) $r = 0b01010101 ^ 0b10101010;

d) $r = 0b00000001<<1;

e) $r = (0b00000001<<1)<<1;

f) $r = ((0b00000001<<1)<<1)<<1;

g) $r = 0b00000001<<3;

h) $r = 0x80000000>>4;
```

```php
u) $r = 0x8000000000000000;

v) $r = (int) 0xffffffffffffffff;

w) $r = (0x80000000 + 0x0fffffff);

x) $r = (0x80000000 + 0x0fffffff)|0;

y) $r = (int) (23 / 2);

z) $r = (int) (23 % 2);

aa) $r = ~false;

bb)   $r = !false;
```

```
i) echo PHP_INT_MAX . PHP_EOL;

j) echo PHP_INT_MIN . PHP_EOL;

k) echo PHP_INT_SIZE . PHP_EOL;

l) $r = 0xffffffff;

m) $r = 0xfffffffff;

n) $r = 0xffffffffffff;

o) $r = 0xfffffffffffff;

p) $r = 0xffffffffffffff;

q) $r = 0x7fffffffffffff;

r) $r = 0x7fffffffffffff<<1;

s) $r = 0x7fffffffffffff<<1 | 1;

t) $r = 0xffffffffffffffff;
```

```
cc)   $r = !true;

dd)   $a = 10;
      $r = ~$a === -$a - 1;

ee)   $r = (-0)*10;

ff)   $r = ~0;

gg)   $r = ~~(0);

hh)   $r = -0<<2>>2;

ii)   $r = -1>>1;

jj)   $r = -1>>6;

kk)   $r = -1<<63;
```

Forzar aritmética entera

5. Dadas las siguiente expresiónes en PHP, 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) $r = (19 / 2);

b) $r = (int) (19 / 2);

c) $r = (-19 / 2);

d) $r = (int) (-19 / 2);
```

```
e) $r = ((19 / 2) + (23 / 3)) % 5;

f) $r = (( int) (19 / 2) + (int)(23 / 3) ) % 5;

g) $r = (( (( int) (19 / 2)) + (( int) 23 / 3) )) % 5;

h) $r = (int) ((( (( int) (19 / 2)) + (( int) 23 / 3) )) % 5);
```

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 PHP:

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