Ejercicios #1 Programación II

Indicaciones: Desarrolle todos los ejercicios de manera ordenado, legible y con nombres de variables relevantes y con documentación interna. Los programas se agregaran a un repositorio en GitHub con nombre programacion2-ejercicios. Por cada ejercicio deberá crear una carpeta llamada ejercicio1, ejercicio2, ejercicio3, etc. Adicionalmente cada programa deberá tener su diagrama de flujo sencillo hecho en con el programa DIA y exportado como imagen png. Además los programas deberán contener al menos 3 casos de prueba para el programa y las funciones desarrollado en otro archivo.

Ejemplo:

> Escribir una función que dado una lista de números naturales, los ordene de tal forma que muestre el numero mas grande posible. Pj. Dado [50, 2, 1, 9], el número que se puede formar que sea mas grande es el 95021
>
> El nombre la función será: *numero_mayor(array $numeros)* y devolverá un entero.
>
> Solución: Puede ver o descargar el proyecto desde https://github.com/neozerosv/programacion2-ejercicios

1. Escriba un programa que pida al usuario un entero de tres dígitos, y entregue el número con los dígitos en orden inverso. Pj.

```
Ingrese numero: 345  //Devuelve 543
Ingrese numero: 241  //Devuelve 142
```

   Pista: Puede utilizar la función de php readline("Ingrese dato:") para poder capturar una entrada desde modo texto.

2. Escriba un programa que muestre todas las combinaciones posibles al momento de lanzar dos dados de 6 caras.

3. Escriba un programa que reciba como entrada el radio de un círculo y entregue como salida su perímetro y su área. El perímetro esta dado por $P=2\pi r$, y el área de un círculo esta dado por $A=\pi r^2$ donde *r* es el radio.

```
Ingrese el radio: 5
Perimetro: 31.4
Área: 78.5
```

   Pista: En php el valor de pi se puede obtener llamando a la función *pi()* y para obtener la potencia puede usar la función *pow()*.

4. Desarrolle un programa para estimar el valor de π usando la siguiente suma infinita desarrollada po leibniz: $\pi=4*\sum_{n=0}^{n}\frac{(-1)^n}{2n+1}$ (Sumatoria desde n=0 hasta n de 4*[(-1)^n/(2n+1)]).

   Ejemplo de la sumatoria con n=4: $\pi=4*\left(1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}\right)$

La entrada del programa debe ser un número entero n que indique cuántos términos de la suma se utilizará.

```
n: 3 //Devuelve 3.466666666666667
n: 1000 //Devuelve 3.140592653839794
```

Pista: En php para obtener la potencia puede usar la función *pow()*.

5. El método de multiplicación rusa consiste en multiplicar sucesivamente por 2 el multiplicando y dividir por 2 el multiplicador hasta que el multiplicador tome el valor 1. Luego, se suman todos los multiplicandos correspondientes a los multiplicadores impares.

   Dicha suma es el producto de los dos números. La siguiente tabla muestra el cálculo realizado para multiplicar 37 por 12, cuyo resultado final es 12 + 48 + 384 = 444.

Multiplicador	Multiplicando	Multiplicador impar	Suma
37	12	sí	12
18	24	no	
9	48	sí	60
4	96	no	
2	192	no	
1	384	sí	444

   Desarrolle un programa que reciba como entrada el multiplicador y el multiplicando, y entregue como resultado el producto de ambos, calculado mediante el método de multiplicación rusa.

```
Ingrese multiplicador: 37
Ingrese multiplicando: 12
Resultado: 444
```

6. Cuando la Tierra completa una órbita alrededor del Sol, no han transcurrido exactamente 365 rotaciones sobre sí misma, sino un poco más. Más precisamente, la diferencia es de más o menos un cuarto de día.

   Para evitar que las estaciones se desfasen con el calendario, el calendario juliano introdujo la regla de introducir un día adicional en los años divisibles por 4 (llamados bisiestos), para tomar en consideración los cuatro cuartos de día acumulados.

   Sin embargo, bajo esta regla sigue habiendo un desfase, que es de aproximadamente 3/400 de día.

   Para corregir este desfase, en el año 1582 el papa Gregorio XIII introdujo un nuevo calendario, en el que el último año de cada siglo dejaba de ser bisiesto, a no ser que fuera divisible por 400.

   Escriba un programa que indique si un año es bisiesto o no, teniendo en cuenta cuál era el calendario vigente en ese año:

```
Ingrese un anio: 1988
1988 es bisiesto

Ingrese un anio: 2011
2011 no es bisiesto

Ingrese un anio: 1700
1700 no es bisiesto

Ingrese un anio: 1500
1500 es bisiesto

Ingrese un anio: 2400
2400 es bisiesto
```

7. Desarrollar una programa que agregue al azar 10 minas en una cuadricula de 20x20 para el juego buscaminas. Esta cuadricula deberá mostrarse en pantalla, mostrando (*) para minas y el (.) para espacios en blanco. La cuadricula de 20x20 deberá almacenarse en un arreglo. Puede usar la función *rand (0,19)* para generar un numero aleatorio entre 0 y 19 (Incluye el 19).

8. Desarrollar una función que reciba como parámetro un numero entero y que a partir de el cree un cuadrado de asteriscos con ese tamaño. Los asteriscos solo se verán en el borde del cuadrado, no en el interior.

9. Escribir una función recursiva que devuelva el factorial de un numero. Un número factorial de un numero n, es el producto de todos los enteros positivos entre 1 y n (incluye n). Por ejemplo el factorial de 4 es 1x2x3x4=24, el factorial de 5 es 1x2x3x4x5=120. Esto puede expresarse recursivamente de la siguiente forma:
   (a) Si n==0, devolver 1. (Este es el caso base)
   (b) Si n>0, se calcula el factorial de n-1, multiplicado por n, y devuelve el resultado.

   El nombre la función será: *factorial(int $numero)* y devolverá un entero.

10. Desarrollar una función que reciba un string (arreglo de caracteres) como parámetro y cuente cuantas vocales tiene esa cadena de caracteres.

    El nombre la función será: *vocales(string $oracion)* y devolverá un entero.

11. Desarrollar una función que verifique si un numero es primo o no. Un numero primo es aquel numero natural que solo tiene dos divisores, el mismo y uno.

    El nombre la función será: *primo(int $numero)* y devolverá un entero.

12. Desarrollar una función (no usar funciones predefinidas de php) que cuente cuantos caracteres tiene un string.

    El nombre la función será: *contar_caracteres(string $oracion)* y devolverá un entero.

13. Desarrollar una función que convierta números romanos a números arábigos.

Los números romanos aún son utilizados para algunos propósitos.

Los símbolos básicos y sus equivalencias decimales son:

M	1000
D	500
C	100
L	50
X	10
V	5
I	1

Los enteros romanos se escriben de acuerdo a las siguientes reglas:

Si una letra está seguida inmediatamente por una de igual o menor valor, su valor se suma al total acumulado. Así, XX = 20, XV = 15 y VI = 6.
Si una letra está seguida inmediatamente por una de mayor valor, su valor se sustrae del total acumulado. Así, IV = 4, XL = 40 y CM = 900.

Escriba la función romano_a_arabigo que reciba un string con un número en notación romana, y entregue el entero equivalente:

```
romano_a_arabigo('MCMXIV') //Devuelve 1914
romano_a_arabigo('XIV')//Devuelve 14
romano_a_arabigo('X') //Devuelve 10
romano_a_arabigo('IV') //Devuelve 4
romano_a_arabigo('DLIV') //Devuelve 554
romano_a_arabigo('CCCIII') //Devuelve 303
```

El nombre la función será: *romano_a_arabigo(string $numRomano)* y devolverá un entero.

14. Desarrollar un programa que calcule el precio para enviar un mensaje por telégrafo. Para esto se sabe que cada letra cuesta $10, los caracteres especiales que no sean letras cuestan $30 y los dígitos tienen un valor de $20 cada uno. Los espacios no tienen valor.

    Su mensaje debe ser un string, y las letras del castellano (ñ, á, é, í, ó, ú) se consideran caracteres especiales.

    ```
    Mensaje: Feliz Aniversario!
    Su mensaje cuesta $190
    ```

15. Desarrollar una función que calcule la raíz cuadrada aproximada de acuerdo a lo siguiente:

    Se toma el número inicial y se le resta el primer número impar (el uno), a este resultado se le resta el siguiente número impar y así sucesivamente hasta que el resultado de la resta sea menor o igual a cero.

Si el resultado final es igual a cero se trata de un número con raíz entera y estará dada por la cantidad de veces que se hizo la resta, incluyendo el cero.
Si el resultado es menor que cero, el número no tiene raíz perfecta y el resultado aproximado (truncado) estará dada por la cantidad de veces que se hizo la resta menos uno.

Por ejemplo:

36

36 - 1 = 35
35 - 3 = 32
32 - 5 = 27
27 - 7 = 20
20 - 9 = 11
11 - 11 = 0

6 veces
raíz aproximada: 6

8

8 - 1 = 7
7 - 3 = 4
4 - 5 = -1

3 veces
raíz aproximada: 3

La dinámica de la función deberá ser la siguiente:

```
raiz_aproximada(25) //Muestra 5
raiz_aproximada(6)  //Muestra 2
raiz_aproximada(1)  //Muestra 1
```

El nombre la función será: *raiz_aproximada(int $numero)* y devolverá un entero.

16. Crear una clase *Calculadora* que almacene dos valores, luego los sume, reste, multiplique, divida entre ellos de acuerdo a la operación que se le solicite. Por ejemplo:

```
$calc = new Calculadora( 3, 4 );
echo $calc- > suma(); // Muestra “7”
echo $calc- > multiplica(); // Muestra “12”
```