## DEL 3 AL 16 DE MAYO

### Ejercicio 11

Crea una biblioteca de funciones matemáticas que contenga las siguientes funciones. Recuerda que puedes usar unas dentro de otras si es necesario. Observa bien lo que hace cada función ya que, si las implementas en el orden adecuado, te puedes ahorrar mucho trabajo. Por ejemplo, la función esCapicua resulta trivial teniendo voltea y la función siguientePrimo también es muy fácil de implementar teniendo esPrimo.

1. esCapicua: Devuelve verdadero si el número que se pasa como parámetro es capicúa y falso en caso contrario.
2. esPrimo: Devuelve verdadero si el número que se pasa como parámetro es primo y falso en caso contrario.
3. siguientePrimo: Devuelve el menor primo que es mayor al número que se pasa como parámetro.
4. potencia: Dada una base y un exponente devuelve la potencia.
5. digitos: Cuenta el número de dígitos de un número entero.
6. voltea: Le da la vuelta a un número.
7. digitoN: Devuelve el dígito que está en la posición n de un número entero. Se empieza contando por el 0 y de izquierda a derecha.

### Ejercicio 12

Amplía el ejercicio anterior con las siguientes funciones:

8. posicionDeDigito: Da la posición de la primera ocurrencia de un dígito dentro de un número entero. Si no se encuentra, devuelve -1.
9. quitaPorDetras: Le quita a un número n dígitos por detrás (por la derecha).
10. quitaPorDelante: Le quita a un número n dígitos por delante (por la izquierda).
11. pegaPorDetras: Añade un dígito a un número por detrás.
12. pegaPorDelante: Añade un dígito a un número por delante.
13. trozoDeNumero: Toma como parámetros las posiciones inicial y final dentro de un número y devuelve el trozo correspondiente.
14. juntaNumeros: Pega dos números para formar uno.

### Ejercicio 13

Crea la función de manejo de arrays que tenga la siguiente cabecera y que haga lo que se especifica en los comentarios (puedes incluirla en tu propia biblioteca de rutinas):

```
public static int[] filtraPrimos(int x[])
// Devuelve un array con todos los números primos que se encuentren
// en otro array que se pasa como parámetro. Obviamente el tamaño
```

```
// del array que se devuelve será menor o igual al que se pasa como
// parámetro.
```

Utiliza esta función en un programa para comprobar que funcionan bien. Para que el ejercicio resulte más fácil, las repeticiones de primos se conservan; es decir, si en el array x el número 13 se repite 3 veces, en el array devuelto también estará repetido 3 veces. Si no existe ningún número primo en x, se devuelve un array con el número -1 como único elemento.

## Ejercicio 14

Crea una función con la siguiente cabecera:

```
public String convierteEnMorse(int n)
```

Esta función convierte el número n al sistema Morse y lo devuelve en una cadena de caracteres. Por ejemplo, el 213 es el . . _ _ _ . _ _ _ _ . . . . _ _ en Morse. Utiliza esta función en un programa para comprobar que funciona bien. Desde la función no se debe mostrar nada por pantalla, solo se debe usar print desde el programa principal.

```
1 . _ _ _ _ 6 _ . . . . .
2 . . _ _ _ 7 _ _ . . .
3 . . . _ _ 8 _ _ _ . .
4 . . . . _ 9 _ _ _ _ .
5 . . . . . 0 _ _ _ _ _
```

## Ejercicio 15

Crea la función de manejo de arrays que tenga la siguiente cabecera y que haga lo que se especifica en los comentarios (puedes incluirla en tu propia biblioteca de rutinas):

```
public int[] filtraCapicuas(int x[])
// Devuelve un array con todos los números capicúa que se encuentren
// en otro array que se pasa como parámetro. Obviamente el tamaño del
// array que se devuelve será menor o igual al que se pasa como
//parámetro.
```

Utiliza esta función en un programa para comprobar que funcionan bien. Para que el ejercicio resulte más fácil, las repeticiones de números capicúa se conservan; es decir, si en el array x el número 505 se repite 3 veces, en el array devuelto también estará repetido 3 veces. Si no existe ningún número capicúa en x, se devuelve un array con el número -1 como único elemento.

## Ejercicio 16

Realiza un programa que pinte un triángulo hueco tal como se muestra en los ejemplos. El usuario debe introducir la altura de la figura. Utiliza funciones para pintar las líneas.

Ejemplo 1:

```
Introduzca la altura de la figura: 8
********
*      *
*     *
*    *
*   *
*  *
* *
**
*
```

Ejemplo 2:

```
Introduzca la altura de la figura: 5
*****
*   *
*  *
* *
**
*
```

## Ejercicio 17

Define la función convierteArrayEnString con la siguiente cabecera:
public static String convierteArrayEnString(int[] a)
Esta función toma como parámetro un array que contiene números y devuelve una cadena de caracteres con esos números. Por ejemplo, si a = { }, convierteArrayEnString(a) devuelve “”; si a = { 8 }, convierteArrayEnString(a) devuelve “8”; si a = { 6, 2, 5, 0, 1 }, convierteArrayEnString(a) devuelve “62501”.

## Ejercicio 18

Realiza un programa que rellene un array con 10 números aleatorios comprendidos entre 2 y 100 (ambos incluidos) y que los muestre por pantalla. A continuación, el programa indicará para cada uno de ellos si es un número primo y/o un capicúa de la forma que muestra el ejemplo.

Ejemplos:

```
Array generado:
19 22 57 11 3 52 32 46 2 14
El 19 es primo y no es capicúa.
El 22 no es primo y es capicúa.
El 57 no es primo y no es capicúa.
```

El 11 es primo y es capicúa.
El 3 es primo y es capicúa.
El 52 no es primo y no es capicúa.
El 32 no es primo y no es capicúa.
El 46 no es primo y no es capicúa.
El 2 es primo y es capicúa.
14 no es primo y no es capicúa.

## Ejercicio 19

Implementa una función con nombre nEsimo que busque el número que hay dentro de un array bidimensional en la posición n-ésima contando de izquierda a derecha y de arriba abajo, como si se estuviera leyendo. El primer elemento es el 0. Si la posición donde se busca no existe en el array, la función debe devolver -1. Se debe entregar tanto el código de la función como el código de prueba que la usa. La cabecera de la función es la siguiente:

```
public static int nEsimo(int[][] n, int posicion)
```

Si el array a es el que se muestra a continuación:

```
35 72 24 45 42 60
32 42 64 23 41 39
98 45 94 11 18 48
12 34 56 78 90 12
```

```
nEsimo(a, 0) devuelve 35
nEsimo(a, 2) devuelve 24
nEsimo(a, 5) devuelve 60
nEsimo(a, 6) devuelve 32
nEsimo(a, 21) devuelve 78
nEsimo(a, 23) devuelve 12
nEsimo(a, 24) devuelve -1
nEsimo(a, 100) devuelve -1
```

## Ejercicio 20

Crea las funciones cuyas cabeceras se muestran a continuación, observa que tienen el mismo nombre:

```
public static int ocurrencias(int digito, int n)
public static int ocurrencias(int digito, int[] a)
```

La función ocurrencias devuelve el número de veces que aparece un dígito dentro de un número (primera función) o bien el número de veces que aparece un dígito en una serie de números contenidos en un array (segunda función).

Ejemplos:

```
ocurrencias(8, 4672) devuelve 0
```

ocurrencias(5, 25153) devuelve 2
ocurrencias(2, 123456) devuelve 1

Si a = {714, 81, 9, 11}, ocurrencias(1, a) devuelve 4
Si a = {42, 13, 12345, 4}, ocurrencias(4, a) devuelve 3
Si a = {6, 66, 666}, ocurrencias(6, a) devuelve 6

Utiliza estas funciones en un programa para comprobar que funcionan bien.