10.8 Problemas propuestos

1. Mostrar la evolución de la pila de llamadas para el siguiente método, suponiendo que la llamada inicial es `fact(5)`. Tomar como referencia las líneas con `//****`

```
/** n >= 0. */
public static int fact(int n) {
    int p;
    //****
    if (n == 0) { p = 1; }
    else { p = n * fact(n - 1); }
    //****
    return p;
}
```

2. Escribir un método de clase recursivo que muestre en pantalla los números naturales del 1 al n, donde $n > 0$ es el valor pasado como parámetro en la llamada inicial.

3. Escribir un método de clase recursivo que muestre en pantalla los números naturales del n al 1, donde $n > 0$ es el valor pasado como parámetro en la llamada inicial.

4. Escribir qué se muestra por pantalla al ejecutar este programa:

```
public class Raro {
    /** n >= 0. */
    public static void escribeRaro(int n) {
        if (n > 0) {
            System.out.print(n);
            escribeRaro(n - 1);
            System.out.print(n);
        }
        else { System.out.print(0); }
    }

    public static void main(String[] args) { escribeRaro(5); }
}
```

5. Escribir un método de clase recursivo que, dados dos números naturales $a \geq 0$ y $b > 0$, calcule el cociente de su división entera, basándose en el hecho de que dicha operación se puede realizar como una serie de restas sucesivas (de forma similar al problema del cálculo del resto de la división entera de la sección 10.4), siguiendo la recurrencia:

   - $a/b = 0$, si $a < b$,
   - $a/b = (a - b)/b + 1$, si $a \geq b$.

6. Dados dos números enteros a y b, siendo $b \geq 0$, se puede definir recursivamente su producto $a \cdot b$ del modo siguiente:

   - $a \cdot b = 0$, si $b = 0$,
   - $a \cdot b = a \cdot (b - 1) + a$, si $b > 0$.

   Nótese que, para esta definición, la multiplicación se reduce a una secuencia de sumas.

   Considerando que tan sólo es posible utilizar operaciones aditivas, escribir un método de clase recursivo para realizar la operación pedida, siguiendo la definición anterior.

7. Dados dos números enteros a y b, siendo $b \geq 0$, otra forma de definir recursivamente su producto $a \cdot b$ es la siguiente:

   - $a \cdot b = 0$, si $b = 0$,
   - $a \cdot b = (a \cdot 2) \cdot (b/2)$, si $b > 0$ y b es par, y
   - $a \cdot b = (a \cdot 2) \cdot (b/2) + a$, si $b > 0$ y b es impar.

   Este tipo de multiplicación se conoce como *multiplicación a la rusa*, siendo utilizada antiguamente por los comerciantes de dicho país, que no conocían la tabla de multiplicar, para efectuar el producto de dos números positivos cualesquiera utilizando solamente sumas, productos y divisiones por 2.

   Considerando que tan sólo se pueden utilizar productos y divisiones por 2, así como sumas, escribir un método de clase recursivo para realizar la operación pedida, siguiendo la definición anterior.

8. Dados dos números enteros $a \neq 0$ y $b \geq 0$, se puede definir recursivamente la potencia a^b del modo siguiente:

   - $a^b = 1$, si $b = 0$,
   - $a^b = a$, si $b = 1$,
   - $a^b = (a^{b/2}) \cdot (a^{b/2})$, si $b > 1$ y b es par, y
   - $a^b = (a^{b/2}) \cdot (a^{b/2}) \cdot a$, si $b > 1$ y b es impar.

   Considerando que tan sólo se pueden utilizar divisiones por 2 y productos, escribir un método de clase recursivo para realizar la operación pedida, siguiendo la definición anterior.

9. Escribir un método de clase recursivo que devuelva la suma de los dígitos de un número natural $n \geq 0$ pasado como parámetro.

10. Escribir un método de clase recursivo que devuelva el número de dígitos de un número natural $n \geq 0$ pasado como parámetro.

11. Escribir un método de clase recursivo que muestre en orden inverso los dígitos que componen un número natural $n \geq 0$ dado.

12. Escribir un método de clase recursivo que devuelva el valor binario (representado como un entero) de un número natural $n \geq 0$ dado. Por ejemplo, si `n = 5` el método devuelve `101`, pero si `n = 31` el método devuelve `11111`.

13. En un triángulo de Pascal, como se puede observar en el ejemplo de la siguiente figura, cada elemento es la suma de los dos elementos situados sobre él, excepto el primero y último de cada fila que valen 1.

```
                1
             1     1
          1     2     1
       1     3     3     1
    1     4     6     4     1
  1    5    10    10     5     1
1    6    15    20    15     6     1
```

    Escribir un método de clase recursivo que devuelva el i-ésimo elemento de la fila f de un triángulo de Pascal; su cabecera será entonces del tipo `public static int trianguloPascal(int f, int i)`.

    Además, escribir una clase Java que lea de teclado un número de filas y, usando el método diseñado, imprima (no necesariamente formando un triángulo) todos los elementos del correspondiente triángulo de Pascal.

14. Escribir qué se muestra por pantalla al ejecutar este programa:

```
public class Suma{
    public static int sumav(int[] v, int i, int x) {
        int suma = 0;
        if (i < 0) { return suma; }
        if (v[i] == x) { return suma; }
        for (int j = 0; j <= i; j++) { suma = suma + v[j]; }
        System.out.println("Suma parcial: " + suma);
        return suma + sumav(v, i - 1, x);
    }

    public static void main(String[] args) {
        int[] v = {1, 2, 3, 4, 5};
        System.out.println("Suma total: " + sumav(v, 4, 2));
    }
}
```

15. Indicar qué se muestra por pantalla al ejecutar este código:

```
public class Traza{
    public static void main(String[] args) {
        int[] v = {0, 11, 2, 13, 4, 5, 6, 17, 8};
        f(v, 0);
    }

    public static void f(int[] v, int i) {
        if (i >= v.length) {
            System.out.println("-------------");
        }
        else if (i == v[i]) {
            System.out.printf("v[%d]==%d\n", i, v[i]);
            f(v, i + 1);
        }
        else {
            f(v, i + 1);
            System.out.printf("v[%d]!=%d\n", i, i);
        }
    }
}
```

16. Dado un array de enteros `v`, escribir un método de clase recursivo que:
    a) Obtenga la suma de todos los elementos del array.
    b) Obtenga la posición del máximo (mínimo) del array.
    c) Dado un entero `x`, cuente cuántas veces aparece en el array.
    d) Compruebe si el array está ordenado ascendentemente.
    e) Determine la posición del primer (último) elemento no nulo del array.
    f) Determine cuántos ceros consecutivos hay al final del array.
    g) Dadas dos posiciones, `izq` y `der`, del array, `0` $\leq$ `izq` $\leq$ `der` $\leq$ `v.length - 1`, invierta todos los elementos del array situados entre dichas posiciones, esto es, al finalizar la ejecución del método el array contendrá en su posición `izq` el elemento que inicialmente ocupaba la posición `der`, en su posición `izq + 1` el elemento que inicialmente ocupaba la posición `der - 1` y así sucesivamente.
    h) Dadas dos posiciones, `izq` y `der`, del array, `0` $\leq$ `izq` $\leq$ `der` $\leq$ `v.length - 1`, duplique el valor de los elementos del array situados entre dichas posiciones.
    i) Dado un entero `b` > 0, determine si `b` es igual a la suma de todas las componentes de `v`.
    j) Dado un entero `x`, determine la cantidad de elementos del array que son menores que `x`.

k) Determine la cantidad de elementos impares que ocupan posiciones pares del array.

l) Determine la posición, si existe, de la primera subsecuencia del array que comprenda, al menos tres números enteros consecutivos en posiciones consecutivas del array.

17. Escribir la versión recursiva del método de la búsqueda binaria propuesto en el capítulo 9.

18. Dado un array de `String` `v`, escribir un método de clase recursivo que:

    a) determine si es capicúa, esto es, si la primera y última palabra del array son la misma, la segunda y la penúltima palabras también lo son, y así sucesivamente. El método retornará `true` si el array es capicúa o `false` en caso contrario.

    b) dada una palabra `pal`, determine la existencia de dicha palabra en el array entre dos posiciones dadas `ini` y `fin` que cumplen inicialmente: `0` $\leq$ `ini` $\leq$ `fin` $<$ `v.length`. Caso de existir la palabra, el método devolverá la primera posición donde se encuentre la misma, y de no existir el método devolverá el valor -1.

19. Escribir un método de clase recursivo que, datos dos `String` $s1$ y $s2$ y sin hacer uso de los métodos definidos en la clase `String` que resuelven el mismo problema, determine:

    a) si $s2$ es prefijo de $s1$.

    b) si $s2$ es sufijo de $s1$.

    c) si $s2$ es una subcadena de $s1$.

20. Escribir un método de clase recursivo que, dados un `String` s y su longitud l, muestre en orden inverso los caracteres de s.

21. Escribir un método de clase recursivo que dé el mismo resultado que el ejercicio previo pero recibiendo sólo la cadena (`String`). Ayuda: usar el método `substring`.

22. Escribir un método de clase recursivo que compruebe si un `String` s dado es palíndromo. Ayuda: usar el método `substring` para reducir la longitud de s en cada llamada recursiva.