Trabajo Práctico Nº 13
Tema: RECURSIVIDAD

1. Dado el siguiente método recursivo:

```
int puzzle(int base, int limite)
      {
      if(base > limite)
            return -1;
      else
            if(base == limite)
                  return 1;
            else
                  return (base * puzzle(base + 1, limite));
      }
```

a) Identificar:
- el o los casos bases del método puzzle.
- el o los casos generales del método puzzle.

b) Mostrar cuál sería el resultado de las siguientes llamadas al método recursivo:
- **System**.out.println(puzzle(14, 10));
- **System**.out.println(puzzle(4, 7));
- **System**.out.println(puzzle(0, 0));

2. Dado el siguiente método recursivo:

```
int concurso(int base, int limite)
      {
      if(base == limite)
            return 1;
      else
            if(base > limite)
                  return 0;
      return (base + concurso(base + 1, limite));
      }
```

Mostrar lo que retornan las siguientes llamadas:

a) **int** x=concurso(0, 3); x es ....
b) **int** y=concurso(10, 7); y es ....
c) **int** z=concurso(5, 50); z es ....

3. En cada uno de los siguientes métodos identificar: el o los casos bases; una llamada recursiva a una versión más pequeña del método; y explicar que hace cada método.

a)
```
int potencia(int m, int n)
      {
      if(n == 0)
            return 1;
      else
            return (m * potencia(m, n - 1));
      }
```

b)
```
int factorial(int n)
      {
      if(n > 0)
            return (n * factorial(n - 1));
```

```
	else
		if(n == 0)
			return 1;
		else
			return –1;
	}
```

c)
```
void clasificar(int[] datos, int desdeind, int hastaind)
	{
	int maximo;
	if(desdeind != hastaind)
		{
		maximo=posicionMaximoElemento(datos, desdeind, hastaind);
		intercambiar(datos, maximo, hastaind);
		clasificar(datos, desdeind, hastaind - 1);
		}
	}
```

- En el caso del inciso a), ¿qué ocurre cuando *n* es un valor negativo?. Modificarlo.
- En el caso del inciso c), ¿qué ocurre cuando *desdeind* es un valor negativo o *hastaind* es un valor mayor o igual al de la cantidad de elementos en el vector?. Modificarlo.

4. Dado el siguiente método recursivo:
```
		void desconocido(int n, int d)
			{
			int k;
			if(n > 0)
				{
				desconocido(n - 1, d + 1);
				for(k=1; k<=d; k++)
					System.out.println(“ ”);
				for(k=1; k<=d; k++)
					System.out.println(“|”);
				System.out.println(“ ”);
				desconocido(n - 1, d + 1);
				}
			}
```

Resolver los siguientes incisos:
a) ¿Qué hace el método?.
b) Hacer un seguimiento con la llamada desconocido(4, 1).
c) ¿Qué ocurre si *n* es negativo?. Si el método contemplara valores negativos para *n*, ¿sería posible esto?.

5. ¿Cuál es la secuencia numérica generada por los siguientes métodos recursivos?.
a)
```
		int f(int n)
			{
			if(n == 0 || n == 1)
				return 1;
			else
				return (3 * f(n - 2) + 2 * f(n - 1));
			}
```

b)

```
int f(int n)
      {
      if(n == 0 || n == 1)
            return 1;
      else
            return (3 * f(n - 2) + 2);
      }
```

c)

```
int f(int n)
      {
      if(n == 0)
            return 1;
      else
            if(n == 1)
                  return 2;
            else
                  return (2 * f(n - 2) + f(n - 1));
      }
```

6. Escribir un método recursivo que calcule la siguiente suma:

$$1 + 2 + 3 + \dots + (n - 1) + n$$

7. Escribir un método recursivo para encontrar la suma de los enteros pares hasta N.

$$2 + 4 + 6 + \dots + (n - 2) + n$$

8. Escribir un método recursivo para calcular la suma de la siguiente serie de enteros, de 1 hasta N.

$$1^n + 2^{n-1} + 3^{n-2} + \dots + n - 1^2 + n^1$$

9. Suponer que la función G está definida recursivamente de la siguiente forma:

$$G(x, y) = \begin{cases} 1 & \text{sí } x \leq y \\ G(x, y+1) + 1 & \text{sí } y < x \end{cases}$$

Si x e y son enteros positivos.
a) Escribir el método recursivo.
b) Encontrar el valor de G(8, 6).
c) Encontrar el valor de G(100, 10).

10. Sea H(x) un método definido recursivamente $\forall$ x > 0, donde x es un entero:

$$H(x) = \begin{cases} 1 & \text{sí } x = 1 \\ H(x/2)+1 & \text{sí } x > 1 \end{cases}$$

a) Escribir el método recursivo.
b) Encontrar el valor de H(80).
c) ¿Cómo podemos describir lo que hace este método?.

11. Escribir un método recursivo **int** vocales(**String** cd) para calcular el número de vocales de una cadena.

12. A partir de un vector de caracteres, determinar si es un palíndromo. Un palíndromo se cumple si al invertirlo se obtiene la misma secuencia de caracteres. Escribir un método recursivo que retorne TRUE si el vector es un palíndromo y FALSE en caso contrario. Escribir otro método recursivo, sobrecargado, que reciba un String, en vez de un vector de caracteres.

13. Un ejemplo sobre definición recursiva es la definición de multiplicación de números naturales. El producto de a * b donde a y b son enteros positivos, se puede definir como a sumado así mismo un número de veces igual a b. Esta definición es iterativa. Una definición iterativa equivalente sería:

a * b = a	sí b = 1
a * b = a * (b-1) + a	sí b > 1

Con el fin de evaluar 6 * 3 utilizando esta definición, debemos primero evaluar 6 * 2 y entonces al resultado agregarle 6. Para evaluar 6 * 2, debemos primero evaluar 6 * 1 y agregarle 6. Pero 6 * 1 es igual a 6 según la primera parte de la definición. Entonces:

$$6 * 3 = 6 * 2 + 6 = 6 * 1 + 6 + 6 = 6 + 6 + 6 = 18$$

Convertir la definición anterior en un algoritmo recursivo.

14. Convierta el siguiente método iterativo en uno recursivo. El método calcula un valor aproximado de e; la base de los logaritmos naturales, sumando las series:

$$1 + 1/1! + 1/2! + \ldots +1/n!$$

Hasta que los términos adicionales no afecten a la aproximación:

```
double loge()
      {
      double en1, delta, fact;
      int n;
      en1=fact=delta=1.0;
      n=1;
      do
            {
            en1=delta;
            n++;
            fact *= n;
            delta=1.0 / fact;
            }
      while(en1 != en1 + delta);
      return en1;
      }
```

15. El Máximo Común Divisor de dos enteros x e y, se define de la siguiente manera:

maximoComunDivisor(x, y) = y	sí y <= x y x % y = 0
maximoComunDivisor(x, y) = maximoComunDivisor(y, x)	sí x < y
maximoComunDivisor(x, y) = maximoComunDivisor(y, x % y)	para cualquier otro caso

Escribir el método recursivo maximoComunDivisor(x, y).

16. Escribir un método recursivo que liste todos los subconjuntos de dos letras para un conjunto dado de letras.
Ejemplo:

[A, C, E, K] → [A, C], [A, E], [A, K], [C, E], [C, K], [E, K].

17. Definimos combinaciones(n, k) como el número de combinaciones de n elementos agrupados de k en k, es decir, el número de los diferentes grupos que se pueden formar con miembros, dado un conjunto de n miembros para elegir. Así, por ejemplo:

sí n = 4, es decir los elementos {A, B, C, D} entonces combinaciones(4, 2) = 6
el resultado es: (A, B), (A, D), (A, C), (B, C), (B, D), (C, D).

Matemáticamente combinaciones(n, k) se define:

combinaciones(n, 1) = n
combinaciones(n, n) = 1
combinaciones(n, k) = combinaciones(n-1, k-1) + combinaciones(n-1, k) $\forall$ n > k y k > 1

a) Encontrar el valor de combinaciones(8, 5).
b) Escribir un método recursivo para calcular combinaciones(n, k).

18. Realizar un método recursivo que calcule la función de Ackermann definida de la siguiente forma:

A(m, n) = n+1	sí m = 0
A(m, n) = A(m-1, 1)	sí n = 0
A(m, n) = A(m-1, A(m, n-1))	sí m > 0 y n > 0

Escribir el método Ackermann de manera iterativa.

19. Probar el método recursivo del ejercicio 3.c.

20. Sea sec1 y sec2 secuencias de caracteres representadas con arreglos. Escribir un método recursivo que determine si sec2 se encuentra contenido en sec1.

21. Dado un vector de enteros, escribir algoritmos recursivos para:
a) Mostrar el vector de forma inversa.
b) Buscar en el vector un determinado valor.
c) Sumar los elementos del vector.
d) Buscar el valor máximo de los elementos del vector.

22. Escribir un método recursivo que calcule los números de Fibonacci. La serie se representa de la siguiente manera:

$$\text{Fibonacci(n)} = \begin{cases} 1 & \text{sí n = 1, n = 2} \\ \text{Fibonacci(n-1)+ Fibonacci (n-2)} & \text{sí n > 2} \end{cases}$$

Fib(1) = 1
Fib(2) = 1
Fib(3) = Fib(1) + Fib(2)
Fib(4) = Fib(2) + Fib(3)
.
.
.