Soluciones ED2

1. a) El propósito de las pruebas de caja negra o funcionales es comprobar si las salidas que devuelve la aplicación son las esperadas en función de los parámetros de entrada.

   Este tipo de prueba, no consideraría, en ningún caso, el código desarrollado, ni el algoritmo, ni la eficiencia, ni si hay partes del código innecesarias, etc.

   b)

   ```
   if (saldo <= 0)
       System.out.println("Puede realizar una transferencia");
   else if (saldo > 0)
       System.out.println("El saldo es negativo ó 0. No puede realizar
   transferencias")
   ```

2. La complejidad ciclomática o McCabe es un algoritmo que obtiene un grafo por el cual se crean los caminos por donde pasaría la ejecución del programa. El grafo nos dará los caminos mínimos por donde pasará la ejecución por todas las posibilidades de ejecución de un algortimo. A partir del grafo se determina su **complejidad ciclomática**. Es posible hacerlo por tres métodos diferentes, pero todos ellos han de dar el mismo resultado.
   - **V (G) = a - n + 2**, siendo a el número de arcos o aristas del grafo y n el número de nodos.
   - **V (G) = r**, siendo r el número de regiones cerradas del grafo (incluida la externa).
   - **V(G) = c + 1**, siendo c el número de nodos de condición.
3. Es un test parametrizado en el cual se realizarán dos pruebas pasando un String y un número para probar el método `devuelveEnt` de la clase A el cual devuelve el segundo parámetro pasado a la función. Entonces se probará:

   Los valores: "Cadena", 1 y "Cadena", 2. Se crea un objeto miIObjUno y se llama al método con los valores anteriores y `assertEquals()` comprobará si la salida del método `devuelveEnt` es igual a 1. Con lo cual, será un Test Ok en el primer caso y fallido en el siguiente al no ser igual a 1.

4. **Repositorio**: es el lugar de almacenamiento o base de datos de los proyectos.

   Rama: revisiones paralelas del contenido del proyecto para efectuar cambios sin tocar la evolución principal (que normalmente se refiere como rama principapl o trunk).

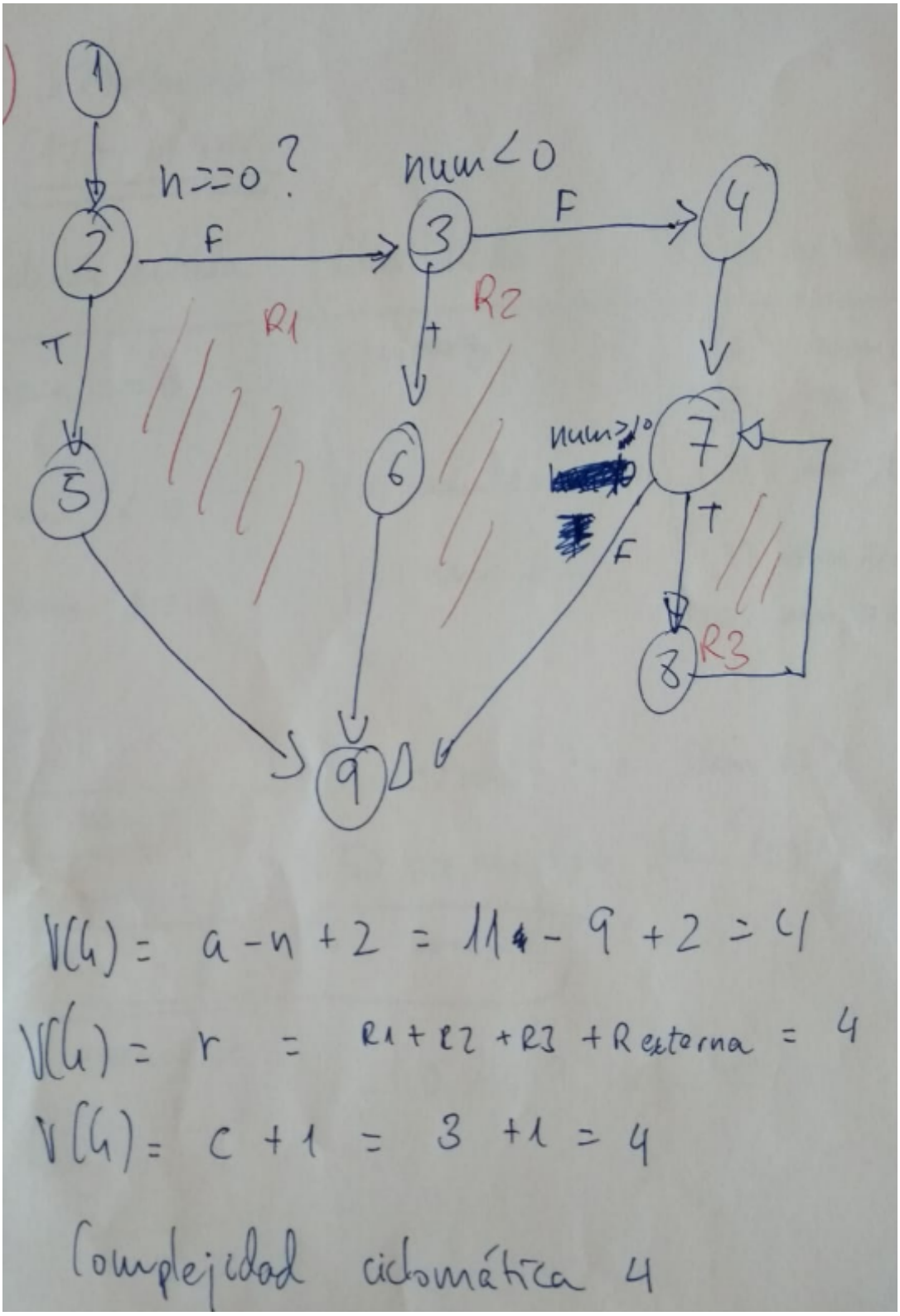
   Versión: fotografía del estado de una de las revisiones de alguna rama del repositorio en un determinado momento en el tiempo.

   Revisión: evolución en el tiempo de las ramas del proyecto.

5. Cambiar de signatura del método es una función dentro de la técnica de Refactorización por el cual se puede renombrar las diversas partes del método con seguridad para el resto de clases: nombre, argumentes, modficadores, etc. Se pueden modificar los modificadores: public, void. El nombre de la función: cambiaSig. El nombre de los parámetros: param1, param2. Y los tipos de los parámetros: double.

```
/**
* @author developer1
* @version 1.1
* @see  http://dev.oracle,com/api11
*/
```

Práctico

1. 

Cambinos de prueba

c1: 1,2,3,4,7,8,7,9

c2:1,2,3,4,7,9

c3; 1,2,3,6,9

c4: 1,2,5,9

Casos de uso | n num num

---

1 |!=0 >=0 > 10

2 |!=0 >=0 <=10

3 |!=0 <0 ---

4 | 0 -- ---

Resultados esperados:

1=> mensaje "Aquí estamos" y calcula ndig

2=> devuelve ndig=1

3=>devuelve -1

4=> "El 0 no cuenta cono número de dígitos" valor=4, Return 0

CAJA NEGRA

======

Condición entrada	**clases válidas**	**Clases no válidas**
num == 0	(1) num = 0	(2) num < 0 (3) num > 0
num < 0	(4) num <0	(5) num >= 0
num > 10	(6) num < 10	(7) num = 10 (8) num > 10

AVL

====

1: num =0

2: num = -1

3: num = 1

4 : num -1

5: num =1

6 : num = 9

7: num = 10

8: num = 10

Como hay casos repetidos escogemos los no repetidos: 0,-1,9,10,20,1 y ya tenemos los AVL

Conjetura de errores:

==========

El valor 0 es un valor como conjetura de error ya previsto anteriormente. También se puede utlizar un carácter o string como entrada, pero no sería totalmente necesario. en este punto.

```
// Pruebas con JUnit
@ParameterizedTest
    @DisplayName("Test de Caja Negra")
    @CsvSource({"0,0","-1,-1","10,1","20,2","1,1"})
    void testNumDigitos(int n, int r) {
        Entornos1 e1 = new Entornos1();
        assertEquals(r, e1.numDigitos(n));
    }
```

2.

```
mkdir -p ~/repoexamen/xED2/src
touch ~/repoexamen/xED2/src/Test.java
cd ~/repoexamen/xED2
git add -all && git commit -m "mensaje" && git push origin master
git log
```

```
# En el repo
git checkout -b secundaria
# Aquí modificamos el fichero Test.java y hacemos git add . && git commit
...etc
git status # sabemos en qué rama estamos
git checkout master # cambiamos a master y podemos ver el fichero Test.java
que con el mensaje antiguo
```