TESTING PROBLEMA 3

1. Código correspondiente a los métodos identificados

```
public class Caso {
    int completitud;
    int correccion;
    int pertinencia;
    int modularidad;
    int reusabilidad;
    int analizabilidad;
    int c_mod;
    int c_prob;

    public Caso() {

    }

 public String funcionamiento(int completitud, int correccion, int pertinencia, int modularidad, int reusabilidad, int analizabilidad, int c_mod, int c_prob) {

      int v_adec = 0;
      int v_mant = 0;

      int [][] matriz_adecuacion = {{0,0,0},{1,1,2},{2,1,2},{2,2,3},{3,3,4},{4,5,5}};
      int [][] matriz_manten = {{0,0,0,0,0},{1,1,0,1,1},{2,2,1,2,1},{3,3,3,4,4},{4,5,5,5,4}};


      int adecuacion [] = {completitud,correccion,pertinencia};
      int mantenibilidad[] = {modularidad,reusabilidad,analizabilidad,c_mod,c_prob};

      adecuacion = devolvervalor(adecuacion,matriz_adecuacion);
      mantenibilidad = devolvervalor(mantenibilidad, matriz_manten);

      System.out.println("\nValores de adecuaciÃ³n: \n");
      for (int i=0; i<adecuacion.length;i++) {
          System.out.print(adecuacion[i]+" ");
      }

      System.out.println("\nValores de mantenimiento: \n");
      for (int i=0; i<mantenibilidad.length;i++) {
          System.out.print(mantenibilidad[i]+" ");
      }


      v_adec = obtenerminimo(adecuacion);
      v_mant = obtenerminimo(mantenibilidad);

      String cadena = "";

      if(!isvalido(v_adec,v_mant))
          cadena = "No cumple alguno de los requisitos para acceder a la prueba de certificado. Valores de adecuaciÃ³n o mantenimiento iguales a 0 o atributos>100 / atributes<0";
      else {
          int [] resultado = obtenercertificado(v_adec, v_mant);
          if(resultado[0]==0)
                cadena="No obtiene certificado. Nivel de calidad: "+  String.valueOf(resultado[1]);
            else
                cadena ="Obtiene certificado de calidad con un nivel de "+ String.valueOf(resultado[1]);

          }

      return cadena;
  }
      public boolean isvalido(int v_adec, int v_mant) {

          boolean valido = false;

          if (v_adec <1 || v_mant < 1)
              valido = false;
          else {
              valido = true;
          }
          return valido;
      }


      public int [] devolvervalor (int [] entradas ,int [][]matriz) {

              int valor, rango;
              int [] resultado = new int [entradas.length];

              for (int i=0; i<entradas.length;i++) {
                  valor = entradas[i];
                  if(valor>=0 && valor<=100) {
                      if(valor < 10)
                          rango = 0;
                      else if (valor>=10 && valor <35)
                          rango = 1;
                      else if (valor >= 35 && valor <50)
                          rango = 2;
                      else if (valor >= 50 && valor<70)
                          rango = 3;
                      else if (valor >= 70 && valor <90)
                          rango = 4;
                      else
                          rango = 5;
                  resultado[i] = rango;
              }
                  else
                      rango = 0;

              }
              return resultado;
      }

      public int obtenerminimo(int[] array) {
          int min = 5;

          for (int i = 0; i < array.length; i++) {

              if(array[i]< min){

                  min=array[i];

              }
          }
          return min;
      }
```

```
public int [] obtenercertificado(int adecuacion_funcional, int mantenibilidad) {
    int [] resultado = {0,0};
    int nivel = 0;

    int certificado;
    int [][] matriz = {{1,1,1,1,1},{1,2,2,2,2},{2,2,3,3,3},{3,3,3,3,4},{3,3,4,4,5}};

    nivel = matriz[adecuacion_funcional-1][mantenibilidad-1];

    if(nivel<3) {
        certificado = 0;
    }
    else {
        certificado = 1;
    }

    resultado[0]=certificado;
    resultado[1]=nivel;

    return resultado;
}
}
```

2. Identificar las variables que se deben tener en cuenta

Hemos seguido el modelo de la tabla, asignando los valores [10,35) al rango 1.

Se deben tener en cuenta las siguientes variables:

Variables que componen la adecuación funcional:

- Completitud: variable entera [0,100] que representa la completitud del producto.
- Corrección: variable entera [0,100] que representa la corrección del producto.
- Pertinencia: variable entera [0,100] que representa la pertinencia del producto.

Variables que componen la mantenibilidad:

- Modularidad: variable entera [0,100] que representa la modularidad del producto.
- Reusabilidad: variable entera [0,100] que representa la reusabilidad del producto.
- Analizabilidad: variable entera [0,100] que representa la analizabilidad del producto.
- C_mod: variable entera [0,100] que representa la capacidad de modificación del producto.
- C_prob: variable entera [0,100] que representa la capacidad de prueba del producto.

3. Indicar los conjuntos de valores de prueba para cada variable

PARÁMETRO	CLASES DE EQUIVALENCIA	VALORES LÍMITE	CONJETURA DE ERRORES
Completitud Funcional	[0,10) [10, 35) [35,50) [50, 70) [70, 90) [90, 100]	0 10 35 50 70 90 100	-1 101
Corrección Funcional	[0,10) [10, 35) [35,50) [50, 70) [70, 90) [90, 100]	0 10 35 50 70 90 100	-1 101
Pertinencia Funcional	[0,10) [10, 35) [35,50) [50, 70) [70, 90) [90, 100]	0 10 35 50 70 90 100	-1 101
Modularidad	[0,10) [10, 35) [35,50) [50, 70) [70, 90) [90, 100]	0 10 35 50 70 90 100	-1 101
Reusabilidad	[0,10) [10, 35) [35,50) [50, 70) [70, 90) [90, 100]	0 10 35 50 70 90 100	-1 101

Analizabilidad	[0,10) [10, 35) [35,50) [50, 70) [70, 90) [90, 100]	0 10 35 50 70 90 100	-1 101
Capacidad de ser Modificado (c_mod)	[0,10) [10, 35) [35,50) [50, 70) [70, 90) [90, 100]	0 10 35 50 70 90 100	-1 101
Capacidad de ser Probado (c_prob)	[0,10) [10, 35) [35,50) [50, 70) [70, 90) [90, 100]	0 10 35 50 70 90 100	-1 101

4. Número máximo de casos de prueba con los valores de prueba acordados

El número máximo de casos de prueba será el producto cartesiano, 9^8, es decir, 43.046.721 casos.

5. Conjunto de casos de prueba para cumplir cobertura each-use

Los casos de prueba serán los siguientes:

Completitud	Corrección	Pertinencia	Modularidad	Reusabilidad	Analizabilidad	Capacidad_serMod	Capacidad_serProb
0	0	0	0	0	0	0	0
10	10	10	10	10	10	10	10
35	35	35	35	35	35	35	35
50	50	50	50	50	50	50	50
70	70	70	90	90	90	90	90
90	90	90	90	90	90	90	90
100	100	100	100	100	100	100	100
-1	-1	-1	-1	-1	-1	-1	-1
101	101	101	101	101	101	101	101

6. Pairwise

Con la gran cantidad de variables y valores para cada variable que tenemos, sería inviable realizar la cobertura pairwise de forma manual ya que son más de 300 combinaciones. Por ello se ha utilizado la herramienta slothman que genera las combinaciones automáticamente.

completitud	correccion	pertinencia	modularidad	reusabilidad	analizabilidad	c_mod	c_prob
0	0	0	0	0	0	0	0
0	10	10	10	10	10	10	10
0	35	35	35	35	35	35	35
0	50	50	50	50	50	50	50
0	70	70	70	70	70	70	70
0	90	90	90	90	90	90	90
0	100	100	100	100	100	100	100
0	-1	-1	-1	-1	-1	-1	-1
0	101	101	101	101	101	101	101
10	10	35	50	70	90	100	-1
10	35	50	70	90	100	-1	101
10	50	70	90	100	-1	101	0
10	70	90	100	-1	101	0	10
10	90	100	-1	101	0	10	35
10	100	-1	101	0	10	35	50
10	-1	101	0	10	35	50	70
10	101	0	10	35	50	70	90
10	0	10	35	50	70	90	100
35	35	70	100	101	10	50	90
35	50	90	-1	0	35	70	100
35	70	100	101	10	50	90	-1
35	90	-1	0	35	70	100	101
35	100	101	10	50	90	-1	0
35	-1	0	35	70	100	101	10
35	101	10	50	90	-1	0	35
35	0	35	70	100	101	10	50
35	10	50	90	-1	0	35	70
50	50	100	0	50	100	0	50
50	70	-1	10	70	-1	10	70
50	90	101	35	90	101	35	90
50	100	0	50	100	0	50	100
50	-1	10	70	-1	10	70	-1
50	101	35	90	101	35	90	101
50	0	50	100	0	50	100	0
50	10	70	-1	10	70	-1	10
50	35	90	101	35	90	101	35
70	70	101	50	-1	35	100	10
70	90	0	70	101	50	-1	35

70	100	10	90	0	70	101	50
70	-1	35	100	10	90	0	70
70	101	50	-1	35	100	10	90
70	0	70	101	50	-1	35	100
70	10	90	0	70	101	50	-1
70	35	100	10	90	0	70	101
70	50	-1	35	100	10	90	0
90	90	10	100	35	-1	50	101
90	100	35	-1	50	101	70	0
90	-1	50	101	70	0	90	10
90	101	70	0	90	10	100	35
90	0	90	10	100	35	-1	50
90	10	100	35	-1	50	101	70
90	35	-1	50	101	70	0	90
90	50	101	70	0	90	10	100
90	70	0	90	10	100	35	-1
100	100	50	0	100	50	0	100
100	-1	70	10	-1	70	10	-1
100	101	90	35	101	90	35	101
100	0	100	50	0	100	50	0
100	10	-1	70	10	-1	70	10
100	35	101	90	35	101	90	35
100	50	0	100	50	0	100	50
100	70	10	-1	70	10	-1	70
100	90	35	101	90	35	101	90
-1	-1	90	50	10	101	100	70
-1	101	100	70	35	0	-1	90
-1	0	-1	90	50	10	101	100
-1	10	101	100	70	35	0	-1
-1	35	0	-1	90	50	10	101
-1	50	10	101	100	70	35	0
-1	70	35	0	-1	90	50	10
-1	90	50	10	101	100	70	35
-1	100	70	35	0	-1	90	50
101	101	-1	100	90	70	50	35
101	0	101	-1	100	90	70	50
101	10	0	101	-1	100	90	70
101	35	10	0	101	-1	100	90
101	50	35	10	0	101	-1	100
101	70	50	35	10	0	101	-1
101	90	70	50	35	10	0	101
101	100	90	70	50	35	10	0
101	-1	100	90	70	50	35	10

7. Cobertura de decisiones

Se realizará una tabla para cada decisión en la que se indicarán únicamente los valores que influyen en dicha decisión para que no quede una tabla demasiado grande.

El cálculo del nivel de calidad se obtiene a partir de los valores de la Adecuación Funcional y la mantenibilidad. Ambas características se obtienen con una función de mínimos.

Adecuación Funcional = Min {Completitud Funcional, Corrección Funcional, Pertinencia Funcional}
Mantenibilidad = Min {Modularidad, Reusabilidad, Analizabilidad, Capacidad de ser modificado, Capacidad de ser probado}

No podemos calcular el nivel de calidad cuando alguno de sus 2 atributos, adecuación funcional o mantenibilidad, son 0.

- (v_adec<1 or v_mant<1)

V_adec	V_mant	Decisión
0	0	T
0	1	T
1	0	T
1	1	F

El método devolverValor es un método genérico con el que obtenemos el valor correspondiente para cada una de nuestras variables. Para cada una de ellas, nos dan un número comprendido entre 0 y 100. Según el rango en el que se encuentre, la variable tendrá un valor de 0 a 5.

En el método devolverValor tenemos todas las decisiones para obtener el valor para cada variable introducida al programa:

- (valor<0 or valor>100)

Valor	Decisión
-1	T
101	T
20	F

- (valor<10)

Valor	Decisión
5	T
11	F

- (valor>=10 and valor <35)

Valor	Decisión
0	F
36	F
20	T

- (valor >= 35 and valor <50)

Valor	Decisión
30	F
55	F

40	T

- (valor >= 50 and valor<70)

Valor	Decisión
45	F
75	F
55	T

- (valor >= 70 && valor <90)

Valor	Decisión
65	F
95	F
75	T

Para que un producto software sea certificable, debe obtener un nivel mayor que 3.

- (nivel<3)

Valor	Decisión
2	T
4	F

8. Cobertura MC/DC

- (v_adec<1 or v_mant<1)

A	B	Decisión	Condición dominante
False (2)	False (2)	False	A,B
False (2)	True (0)	True	B
True (0)	False (2)	True	A
True (0)	True (0)	True	A,B

- (valor<10)

A	Decisión	Condición dominante
True	True	A
False	False	A

- (valor>=10 and valor <35)

A	Decisión	Condición dominante
True	True	A
False	False	A

- (valor >= 35 and valor <50)

A	Decisión	Condición dominante
True	True	A
False	False	A

- (valor >= 50 and valor<70)

A	Decisión	Condición dominante
True	True	A
False	False	A

- (valor >= 70 && valor <90)

A	Decisión	Condición dominante
True	True	A
False	False	A

- (valor > 90)

A	Decisión	Condición dominante
True	True	A
False	False	A

- (valor <0 or valor > 100)

A	Decisión	Condición dominante
True	False	A
False	True	A

- (nivel<3)

A	Decisión	Condición dominante
True	True	A
False	False	A