P02-Diseño de pruebas de caja blanca

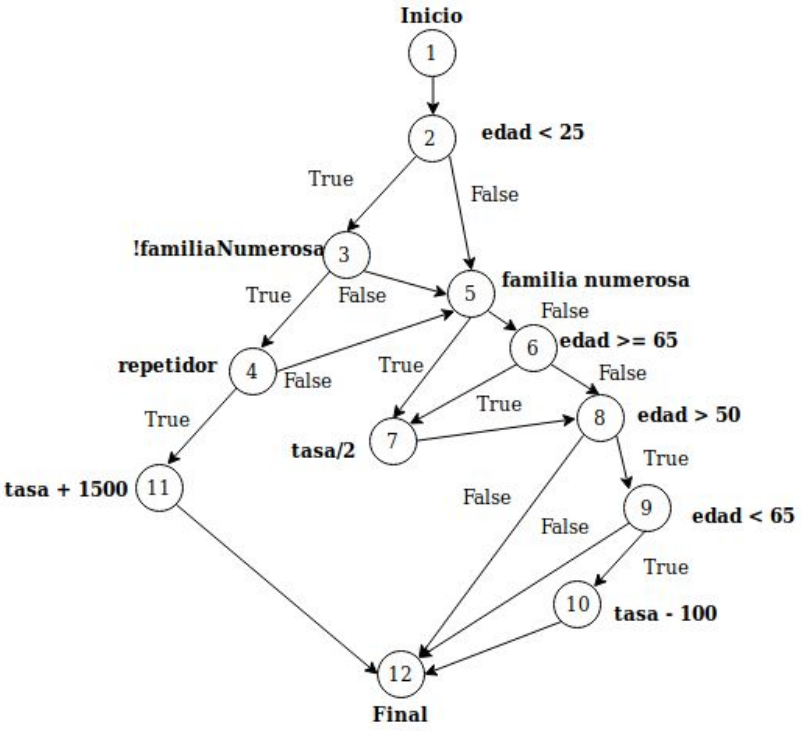
Ejercicio 1: método calcularTasaMatricula()

Queremos diseñar los casos de prueba para el método calculaTasaMatricula() (la especificación es la misma que la del ejercicio 3 de la práctica anterior).

Utiliza el método de casos de prueba del método que hemos visto en clase para diseñar los tests teniendo en cuenta que la implementación es la siguiente:

```
1. public float calculaTasaMatricula(int edad, boolean familiaNumerosa,
2.                                    boolean repetidor) {
3.   float tasa = 500.00f;
4.
5.   if ((edad < 25) && (!familiaNumerosa) && (repetidor)) {
6.     tasa = tasa + 1500.00f;
7.   } else {
8.      if ((familiaNumerosa) || (edad >= 65)) {
9.        tasa = tasa / 2;
10.     }
11.     if ((edad > 50) && (edad < 65)) {
12.       tasa = tasa - 100.00f;
13.     }
14.  }
15.  return tasa;
16.}
```

El grafo es el siguiente:



CC = 18-12+2 -> 8

C1 = 1 - 2 - 3 - 4 - 11

C2 = 1 - 2 - 3 - 5 - 7 - 12

C3 = 1 - 2 - 3 - 4 - 5 - 6 - 8 - 12

C4 = 1 - 2 - 5 - 6 - 7 - 8 - 9 - 10 - 12

C5 = 1 - 2 - 5 - 6 - 7 - 8 - 9 - 12

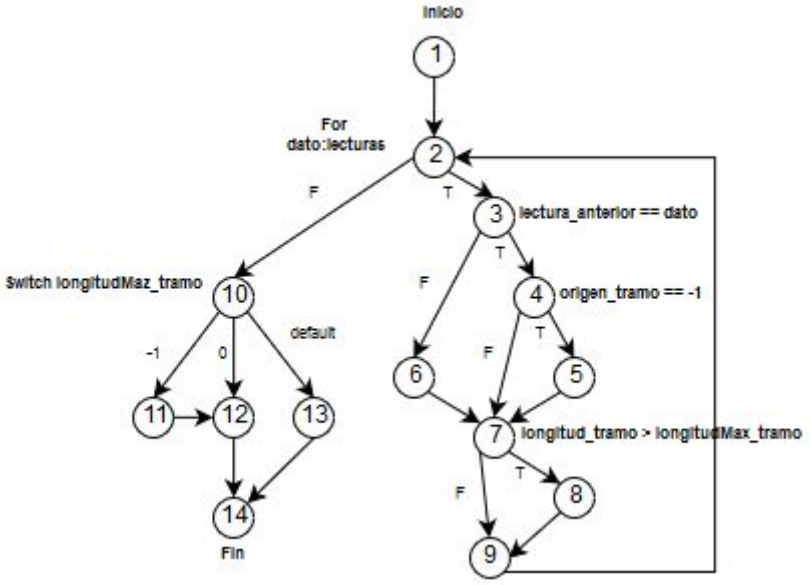
Identificador del Caso de prueba	**Datos de entrada**			**Resultado esperado**
	Edad	**Familia numerosa**	**Repetidor**	
C1	24	false	true	2000
C2	24	true	false	250
C3	24	false	false	500?
C4	55	true	false	250
C5	66	true	false	400

Ejercicio 2: método buscarTramoLlanoMasLargo()

Se proporciona una especificación para el método buscarTramoLlanoMasLargo() al igual que el siguiente código:

```
1.public Tramo buscarTramoLlanoMasLargo(ArrayList<Integer> lecturas) {
2.   int lectura_anterior =-1;
3.   int longitud_tramo =0, longitudMax_tramo=0;
4.   int origen_tramo=-1, origen_tramoMax=-1;
5.   Tramo resultado = new Tramo(); //el origen y la longitud es CERO
6.
7.   for(Integer dato:lecturas) {
8.      if (lectura_anterior== dato) {//detectamos un llano
9.         longitud_tramo ++;
10.         if (origen_tramo == -1 ) {//marcamos su origen
11.             origen_tramo = lecturas.indexOf(dato);
12.         }
13.      } else {  //no es un llano o se termina el tramo llano
14.          longitud_tramo=0;
15.          origen_tramo=-1;
16.      }
17.      //actualizamos la longitud máxima del llano detectado
18.      if (longitud_tramo > longitudMax_tramo) {
19.         longitudMax_tramo = longitud_tramo;
20.         origen_tramoMax = origen_tramo;
21.      }
22.      lectura_anterior=dato;
23.   }
24.   switch (longitudMax_tramo) {
25.     case -1:
26.     case  0: break;
27.     default: resultado.setOrigen(origen_tramoMax);
28.              resultado.setDuracion(longitudMax_tramo);
29.   }
30.
31.   return resultado;
32.}
```

Se pide:
A) Representa el CFG asociado al siguiente código, que implementa esta especificación, calcula su CC, y obtén el conjunto de caminos independientes.



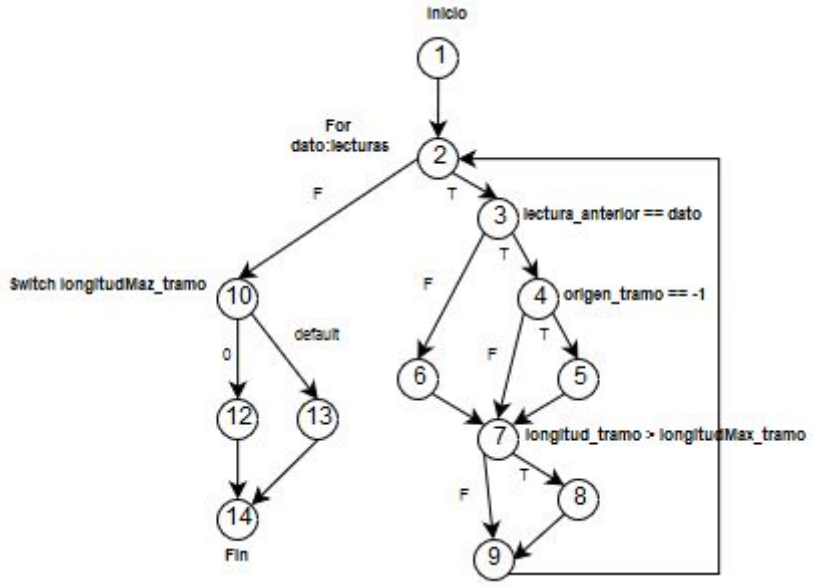
CC = 19 - 14 + 2 -> 7

C1 = 1 - 2 - 3 - 6 - 7 - 9 - 2 - 10 - 12 - 14 {1}
C2 = 1 - 2 - 3 - 6 - 7 - 9 - 2 - 3 - 4 - 5 - 7 - 8 - 9 - 2 - 3 - 4 - 7 - 8 - 9 - 2 - 10 - 13 - 14 {3,3,3}

B) Selecciona datos de entrada para recorrer todos los caminos obtenidos, verás que aparece un camino que es IMPOSIBLE de recorrer con ningún dato de entrada (pista: el problema está en las líneas 24..29). ¿Debemos simplemente ignorar dicho camino?

Deberíamos ignorar dicho camino ya que no va a pasar nunca. LongitudMax_tramo no va a decrementar en ninguna parte del código.

C) Modifica el código para eliminar las sentencias que nunca se va a ejecutar. Representa el nuevo grafo y diseña de nuevo los casos de prueba. ¿Podemos asegurar que el código es correcto? Piensa en algún caso de prueba adicional que pudiera poner de manifiesto un defecto en nuestro código.



Identificador del Caso de prueba	Datos de entrada	Resultado esperado
	Array Lectura	
C1	{1}	0 - 0
C2	{3,3,3}	0 - 3

Ejercicio 3: método realizarLizaReserva()

Se proporciona una especificación para el método realizaReserva() y además el siguiente código:

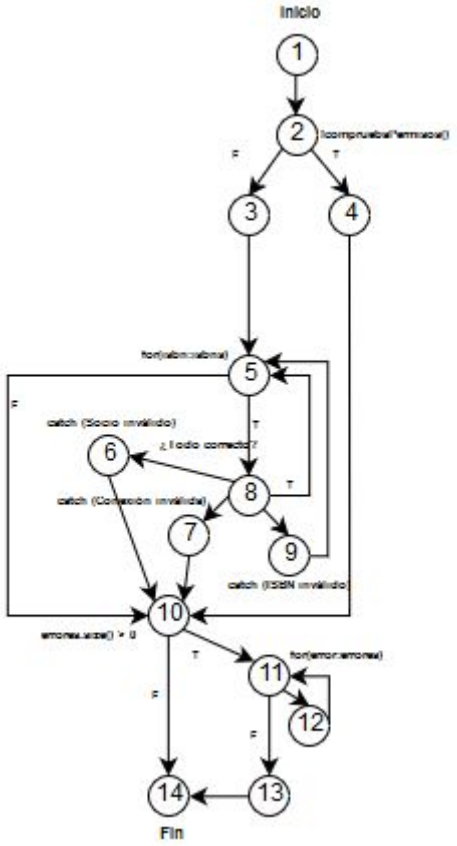
```
1.public void realizaReserva(String login, String password,
2.                           String socio, String [] isbns) throws Exception {
3.
4.  ArrayList<String> errores = new ArrayList<String>();
5.    //El método compruebaPermisos() devuelve cierto si la persona que hace
6.    //la reserva es el bibliotecario y falso en caso contrario
7.  if(!compruebaPermisos(login, password, Usuario.BIBLIOTECARIO)) {
8.    errores.add("ERROR de permisos");
9.  } else {
10.    FactoriaBOs fd = FactoriaBOs.getInstance();
11.    //El método getOperacionBO() devuelve un objeto de tipo IOperacionBO
12.    //a partir del cual podemos hacer efectiva la reserva
13.    IOperacionBO io = fd.getOperacionBO();
14.    try {
15.      for(String isbn: isbns) {
16.        try {
17.          //El método reserva() registra la reserva de un libro
18.          //por parte de un socio, a partir de su identificador e isbn
19.          io.reserva(socio, isbn);
20.        } catch (IsbnInvalidoException iie) {
21.          errores.add("ISBN invalido" + ":" +  isbn);
22.        }
23.      }
24.    } catch (SocioInvalidoException sie) {
25.      errores.add("SOCIO invalido");
26.    } catch (JDBCException je) {
27.      errores.add("CONEXION invalida");
28.    }
29.  }
30.  if (errores.size() > 0) {
31.    String mensajeError = "";
32.    for(String error: errores) {
33.      mensajeError += error + "; ";
34.    }
35.    throw new ReservaException(mensajeError);
36.  }
37.}
```

Figura 1. Implementación del método *realizaReserva()*

A partir del código y la especificación proporcionadas, diseña una tabla de casos de prueba para el método realizaReserva().

CC = 20 - 14 + 2 -> 8

C1 = 1 - 2 - 3 - 5 - 8 - 9 - 5 - 10 - 14	(Todos los datos correctos)
C2 = 1 - 2 - 4 - 10 - 11 - 12 - 11 - 13 - 14	(No sea el bibliotecario)
C3 = 1 - 2 - 3 - 5 - 8 - 6 - 10 - 11 - 12 - 13 -14	(Socio inválido)
C4 = 1 - 2 - 3 - 5 - 8 - 9 - 5 - 10 - 11 - 12 - 11 - 13 - 14	(Isbn Inválido)
C5 = 1 - 2 - 3 - 5 - 8 - 7 - 10 - 11 - 12 - 11 - 13 - 14	(Conexión inválida)



Identificador del Caso de prueba	Datos de entrada				Resultado esperado
	Login	**Password**	**Socio**	**ISBN**	
C1	biblio	biblio	socio	{12345}	(Nada)
C2	*nbiblio	*nbiblio	socio	{12345}	“ERROR de permisos”
C3	biblio	biblio	*nsocio	{12345}	“SOCIO inválido”
C4	biblio	biblio	socio	*{23456}	“ISBN inválido: 23456”
C5	biblio	biblio	socio	{12345}	“CONEXIÓN inválida”

*(Suponemos que el ISBN “23456” es incorrecto)
*(Suponemos que el login y password “nbiblio” es incorrecto)
*(Suponemos que el socio“nsocio” es incorrecto)
*(Todo lo demás es correcto)