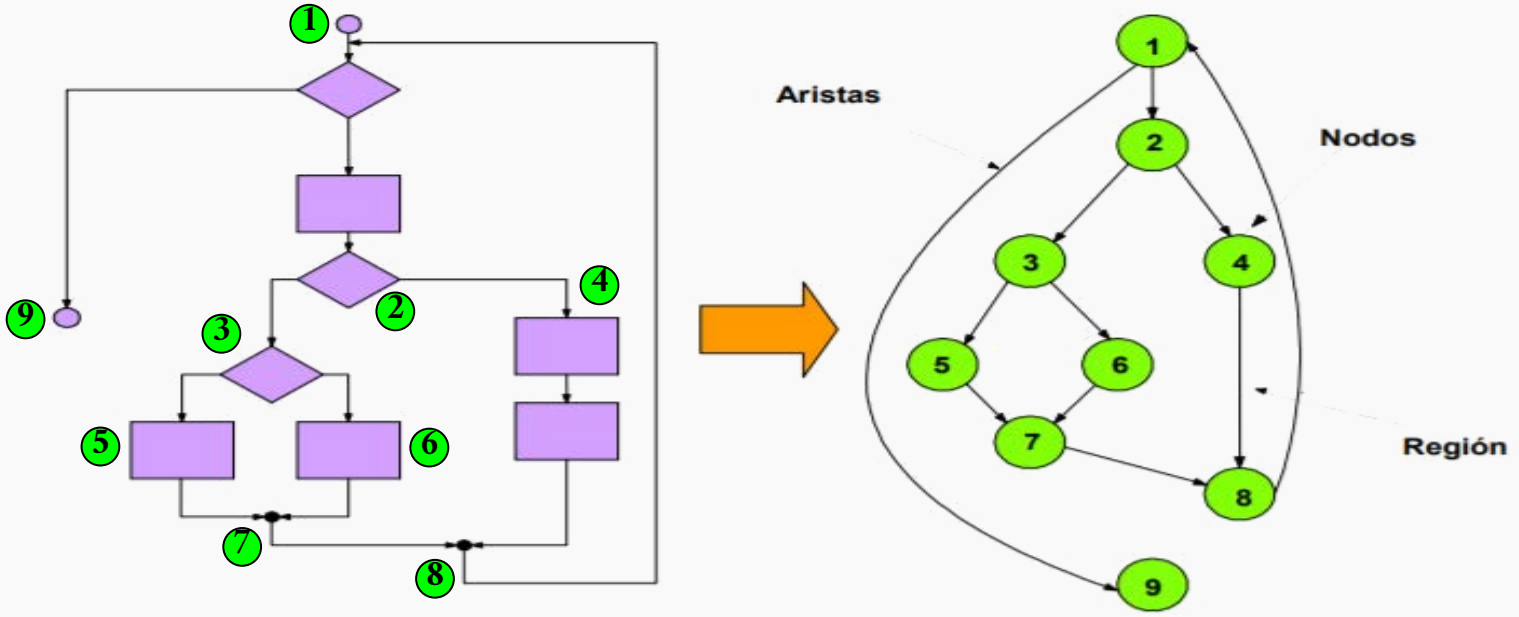
Pruebas Del Software

TEMA 2.1

Pruebas de Caja Blanca.

0. Obtener el grafo de flujo, la complejidad ciclomática, los caminos independientes, y los casos de prueba con cobertura de decisiones,condiciones, y condiciones/decisiones, de algoritmos correspondientes a diagramas de flujo y pseudocódigos.

1. Obtener el grafo de flujo, la complejidad ciclomática, y los caminos independientes, del algoritmo correspondiente al siguiente diagrama de flujo.

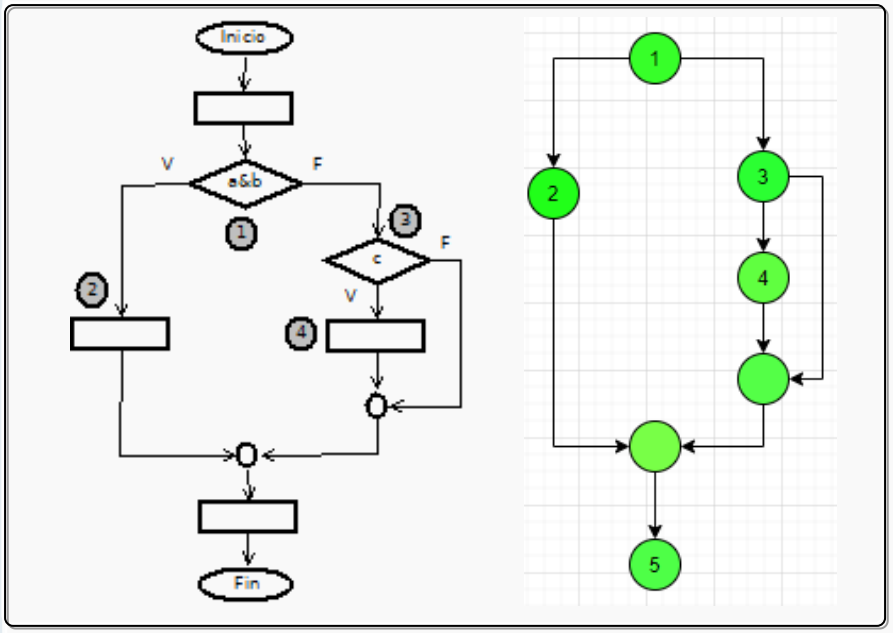


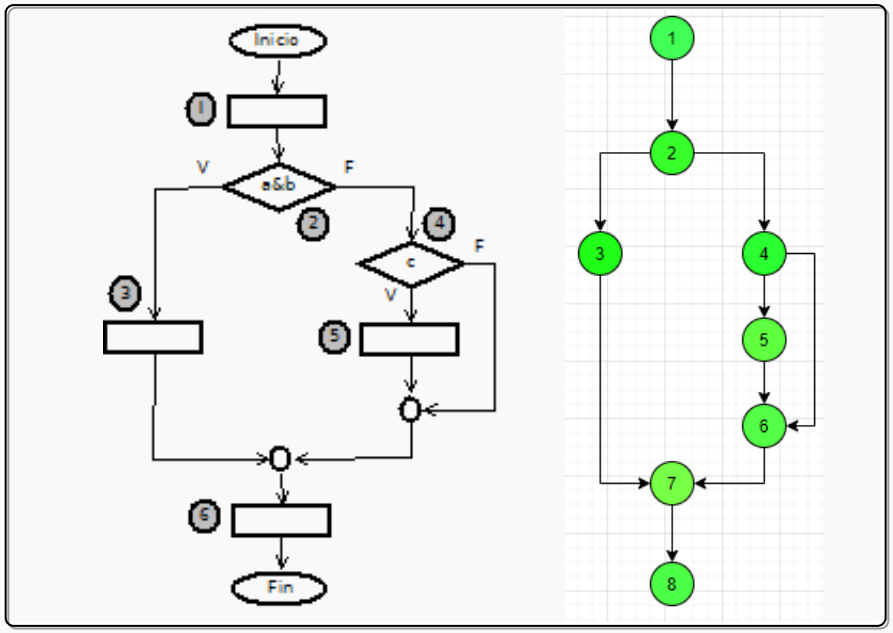
V(G) = número de aristas – número de nodos + 2 = 11 – 9 + 2 = 4
V(G) = número de nodos predicado (número de bifurcaciones) + 1 = 3 + 1 = 4
V(G) = número de regiones delimitadas por aristas (incluida la exterior) = 4

CAMINOS
1-9
1-2-4-8-1-9
1-2-3-5-7-8-1-9
1-2-3-6-7-8-1-9

Pruebas de Caja Blanca.

2. Obtener el grafo de flujo, la complejidad ciclomática, los caminos independientes, y los casos de prueba con cobertura de decisiones, del algoritmo correspondiente al siguiente diagrama de flujo:



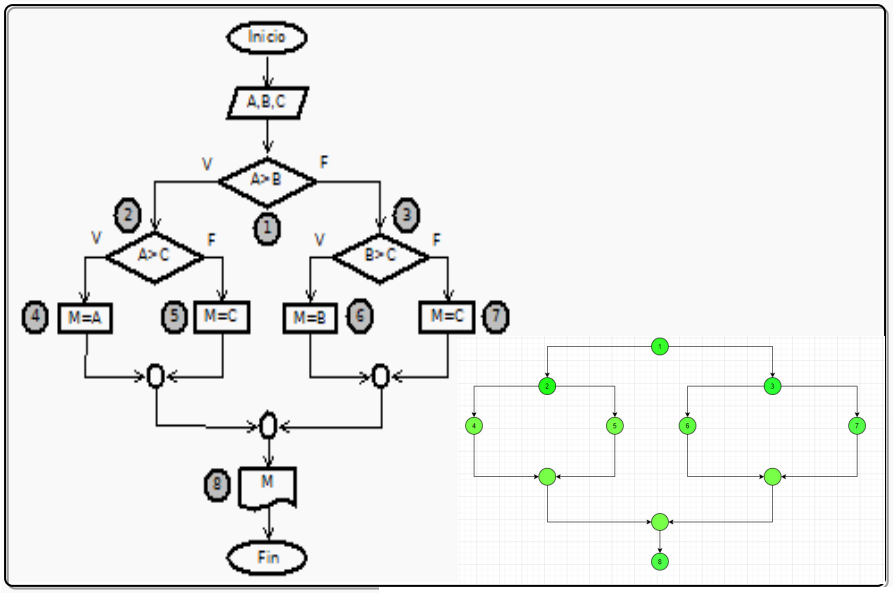


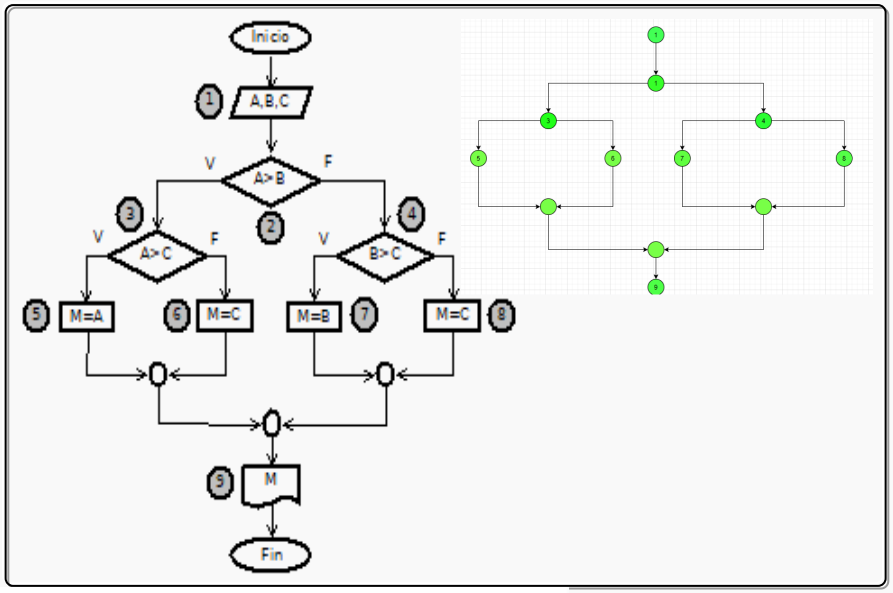
COMPLEJIDAD CICLOMÁTICA: 3

CAMINOS	ENTRADA	SALIDA
1,2,5	A=1,B=1,C=0	¿FIN?
1,3,5	A=0,B=1,C=0	¿FIN?
1,3,4,5	A=0,B=1,C=1	¿FIN?

Pruebas de Caja Blanca.

3. Obtener el grafo de flujo, la complejidad ciclomática, los caminos independientes, y los casos de prueba con cobertura de decisiones, del algoritmo correspondiente al siguiente diagrama de flujo:



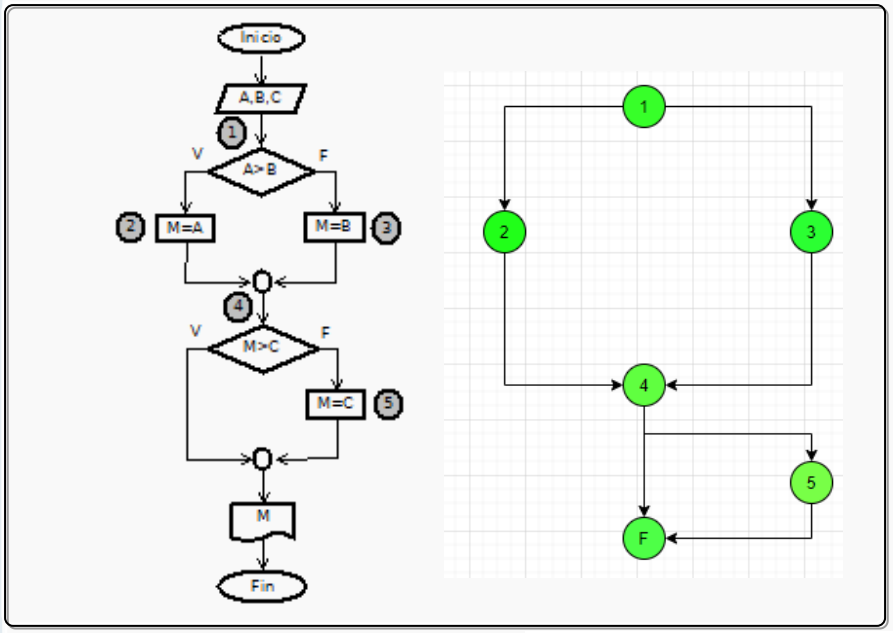


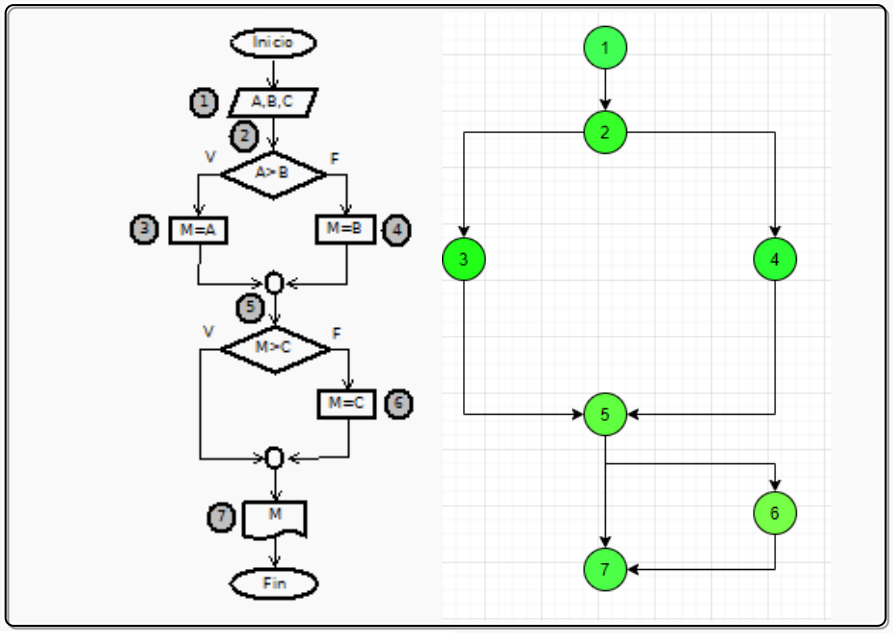
COMPLEJIDAD CICLOMÁTICA: 4

CAMINOS	ENTRADA	SALIDA
1,2,4,8	A=2,B=1,C=0	M=A
1,2,5,8	A=2,B=1,C=3	M=C
1,3,6,8	A=1,B=2,C=0	M=B
1,3,7,8	A=1,B=2,C=3	M=C

Pruebas de Caja Blanca.

4. Obtener el grafo de flujo, la complejidad ciclomática, los caminos independientes, y los casos de prueba con cobertura de decisiones, del algoritmo correspondiente al siguiente diagrama de flujo:





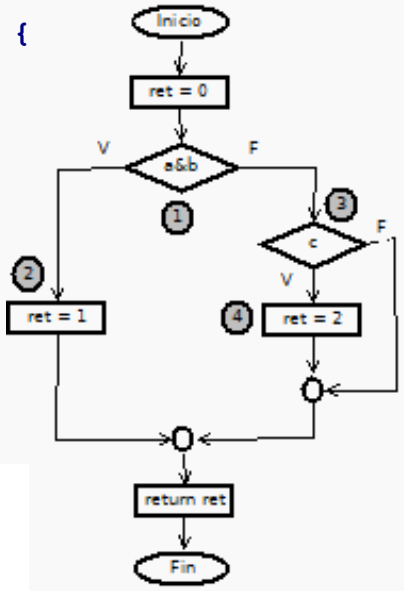
COMPLEJIDAD CICLOMÁTICA: 3

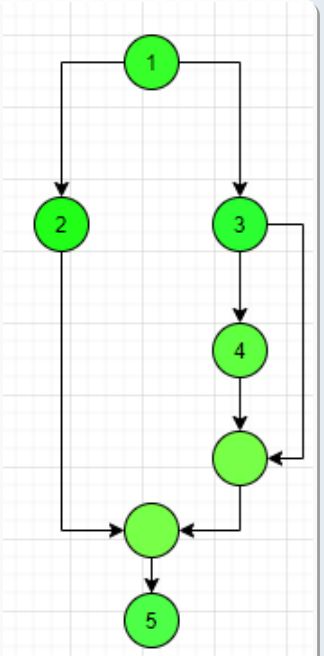
CAMINOS	ENTRADA	SALIDA
1,2,4,F	A=2,B=1,C=0	M=A
1,3,4,F	A=1,B=2,C=1	M=B
1,3,4,5,F	A=1,B=2,C=4	M=C

Pruebas de Caja Blanca.

6a. Obtener el grafo de flujo, la complejidad ciclomática, los caminos independientes, los casos de prueba con cobertura de **decisiones**, y la implementación en JUnit5, del algoritmo correspondiente al siguiente programa.

```
I  Public int aMethod (boolean a, boolean b, boolean c) {
       int ret = 0;
   1   if (a && b) {
   2     ret = 1;
   3     } else if (c) {
   4     ret = 2;
         }
       return ret;
   F   }
```





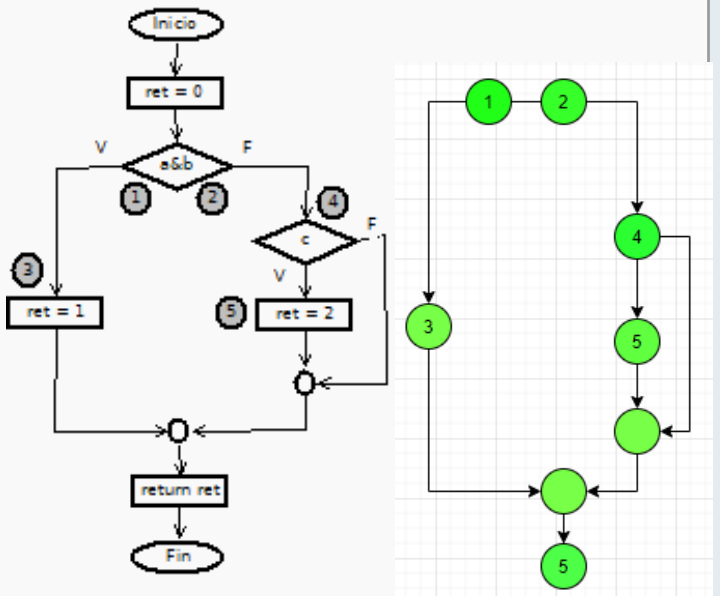
COMPLEJIDAD CICLOMÁTICA: 3

CAMINOS	ENTRADA	SALIDA
1,2,5	A=1,B=1,C=0	M=1
1,3,4,5	A=1,B=1,C=1	M=2
1,3,5	A=1,B=1,C=0	M=0

Pruebas de Caja Blanca.

6b. Obtener el grafo de flujo, la complejidad ciclomática, los caminos independientes, los casos de prueba con cobertura de **condiciones**, y la implementación en JUnit5, del algoritmo correspondiente al siguiente programa.

```
I  Public int aMethod (boolean a, boolean b, boolean c) {
       int ret = 0;
           1    2
       if (a && b) {
         3 ret = 1;
         4 } else if (c) {
         5 ret = 2;
           }
       return ret;
   F }
```



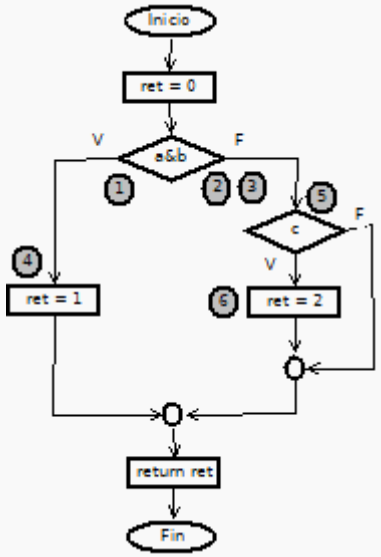
COMPLEJIDAD CICLOMÁTICA: 3

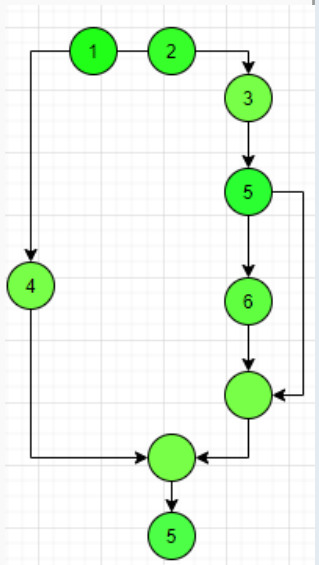
CAMINOS	ENTRADA	SALIDA
1,2,3,F	A=1,B=1,C=0	M=1
1,4,5,F	A=1,B=1,C=1	M=2
1,4,F	A=1,B=1,C=0	M=0

Pruebas de Caja Blanca.

6c. Obtener el grafo de flujo, la complejidad ciclomática, los caminos independientes, los casos de prueba con cobertura de **condiciones/decisiones**, y la implementación en JUnit5, del algoritmo correspondiente al siguiente programa.

```
I  Public int aMethod (boolean a, boolean b, boolean c) {
       int ret = 0;
          1     2 3
       if (a && b) {
         4 ret = 1;
         5 } else if (c) {
         6 ret = 2;
           }
       return ret;
   F }
```





COMPLEJIDAD CICLOMÁTICA: 3

CAMINOS	ENTRADA	SALIDA
1,2,4,F	A=1,B=1,C=0	M=1
1,5,6,F	A=1,B=1,C=1	M=2
1,5,F	A=1,B=1,C=0	M=0

Pruebas de Caja Blanca.

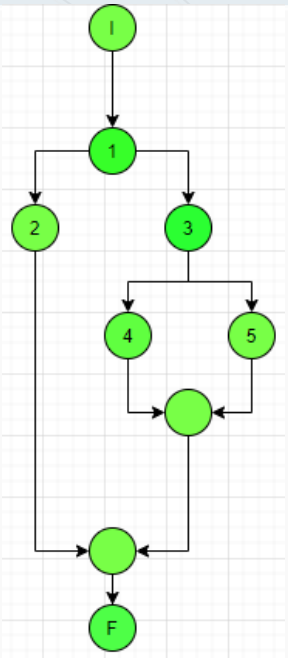
7a. Obtener el grafo de flujo, la complejidad ciclomática, los caminos independientes, los casos de prueba con cobertura de **decisiones**, y la implementación en JUnit5, del algoritmo correspondiente al siguiente programa.

```
import java.util.Scanner;

I public class Maximo {
      public static void main (String args[]) {
            Scanner sc = new Scanner(System.in);

            int x,y,z,max;

            System.out.println("Introduce x,y,z: ");
            x =  sc.nextInt();
            y =  sc.nextInt();
            z =  sc.nextInt();
      1
            if (x>y && x>z)
                  max = x; 2
            else
                  3 if (z>y)
                        max = z; 4
                  else
                        max = y; 5
            System.out.println ("El máximo es " + max);
      }
F }
```



COMPLEJIDAD CICLOMÁTICA: 3

CAMINOS	ENTRADA	SALIDA
1,2,F	X=1,Y=0,Z=0	MAX=X
1,3,4,F	X=0,Y=0,Z=1	MAX=Z
1,3,5,F	X=0,Y=1,Z=0	MAX=1

Pruebas de Caja Blanca.

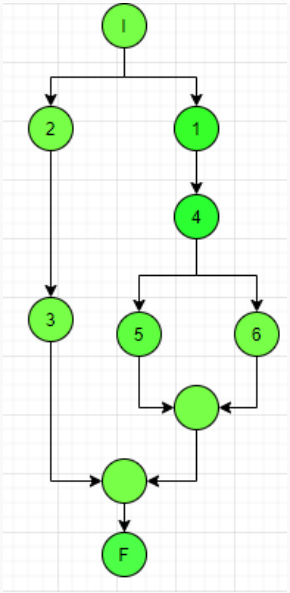
7b. Obtener el grafo de flujo, la complejidad ciclomática, los caminos independientes, los casos de prueba con cobertura de **condiciones**, y la implementación en JUnit5, del algoritmo correspondiente al siguiente programa.

```
import java.util.Scanner;

I public class Maximo {
      public static void main (String args[]) {
            Scanner sc = new Scanner(System.in);

            int x,y,z,max;

            System.out.println("Introduce x,y,z: ");
            x =  sc.nextInt();
            y =  sc.nextInt();
            z =  sc.nextInt();
      1                    2
            if (x>y && x>z)
                  max = x; 3
            else
                  4 if (z>y)
                        max = z; 5
                  else
                        max = y; 6
            System.out.println ("El máximo es " + max);
      }
F }
```



COMPLEJIDAD CICLOMÁTICA: 3

CAMINOS	ENTRADA	SALIDA
I,2,3,F	X=1,Y=0,Z=0	MAX=X
I,1,4,5,F	X=0,Y=0,Z=1	MAX=Z
I,1,4,6,F	X=0,Y=1,Z=0	MAX=1

Pruebas de Caja Blanca.

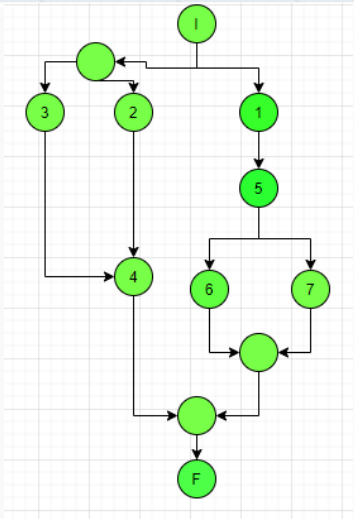
7c. Obtener el grafo de flujo, la complejidad ciclomática, los caminos independientes, los casos de prueba con cobertura de **condiciones/decisiones**, y la implementación en JUnit5, del algoritmo correspondiente al siguiente programa.

```
import java.util.Scanner;

I  public class Maximo {
       public static void main (String args[]) {
            Scanner sc = new Scanner(System.in);

            int x,y,z,max;

            System.out.println("Introduce x,y,z: ");
            x =  sc.nextInt();
            y =  sc.nextInt();
            z =  sc.nextInt();
            if (x>y && x>z)
                  max = x;
            else
                  if (z>y)
                        max = z;
                  else
                        max = y;
            System.out.println ("El máximo es " + max);
       }
F  }
```



COMPLEJIDAD CICLOMÁTICA: 4

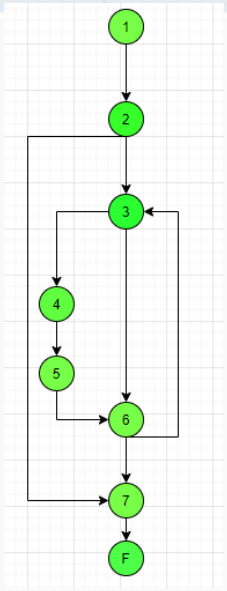
CAMINOS	ENTRADA	SALIDA
I,3,4,F	X=1,Y=0,Z=0	MAX=X
I,2,4,F	X=1,Y=0,Z=0	MAX=X
I,1,5,6,F	X=0,Y=0,Z=1	MAX=Z
I,1,5,7,F	X=0,Y=1,Z=0	MAX=1

Pruebas de Caja Blanca.

8. Obtener el grafo de flujo, la complejidad ciclomática, los caminos independientes, los casos de prueba con cobertura de **segmentos, decisiones y bucles**, y la implementación en JUnit5, del algoritmo correspondiente al siguiente programa.

```
public class PruebaGrafoFlujo {
    public static int contarLetras(String cadena, char letra){
        int contador=0, n=0, lon;
        lon = cadena.length();                    1
    2   if (lon > 0) {
            do {
                if (cadena.charAt(contador)==letra)   3
                    n++;   4
                contador++;
                lon--;                            5
        6   } while(lon>0);
        }
        return n;   7
    }

    public static void main (String args[]) {
        int cuenta = PruebaGrafoFlujo.contarLetras("Hola  mundo mágico",'o');
        System.out.println(cuenta);
    }
}
```



COMPLEJIDAD CICLOMÁTICA: 4

CAMINOS	ENTRADA	SALIDA
1,2,7,F		0
1,2,3,4,5,6,7,F	A	1
1,2,3,6,2,F	HO	2
1,2,3,6,F	1	0