UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL TUCUMÁN

INGENIERÍA DE SOFTWARE

Comisión: 4K3

TRABAJO PRÁCTICO N° 3

Fecha de Presentación: 15/11/23

Grupo N° 20

Integrantes

- Apesoa Martínez, Martín - 48165
- Brito, Patricio Emanuel - 43751
- Morales, Marisol del Valle – 43698
- Gerez, Maria Belén - 44520

INGENIERÍA DEL SOFTWARE

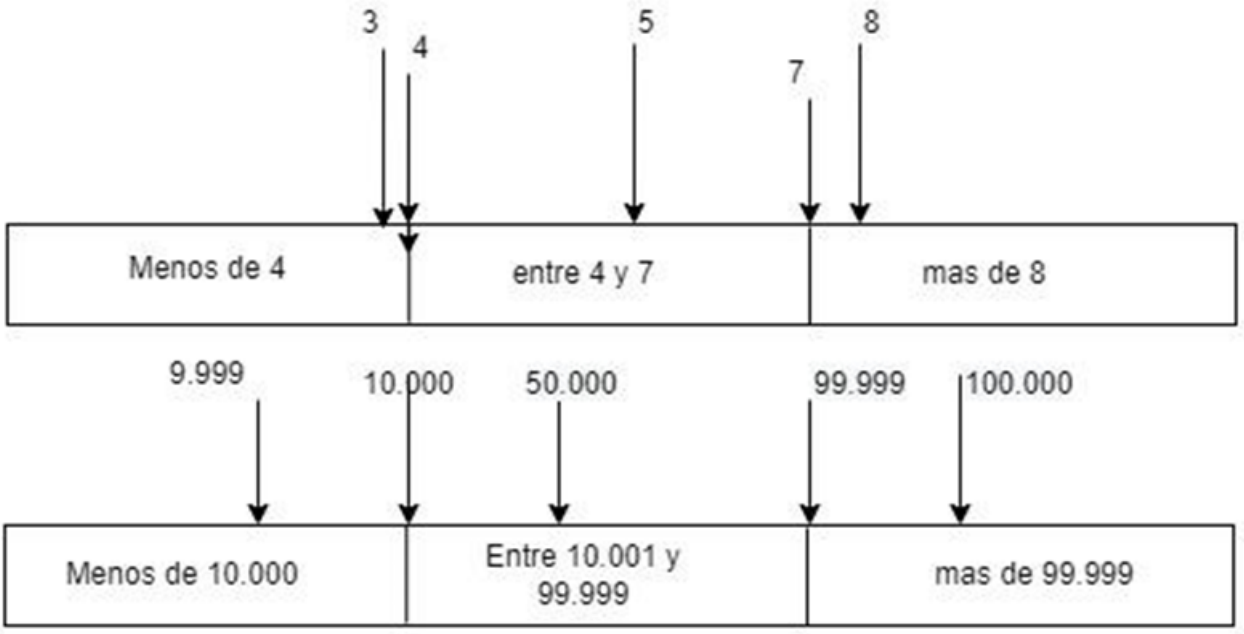
TRABAJO PRÁCTICO N° 3

ESTRATEGIAS PARA CASOS DE PRUEBA, AUTOMATIZACIÓN DE PRUEBAS DE ACEPTACIÓN, PRUEBAS UNITARIAS Y PRUEBAS DE SISTEMA

2023

1. **Pruebas de particiones**

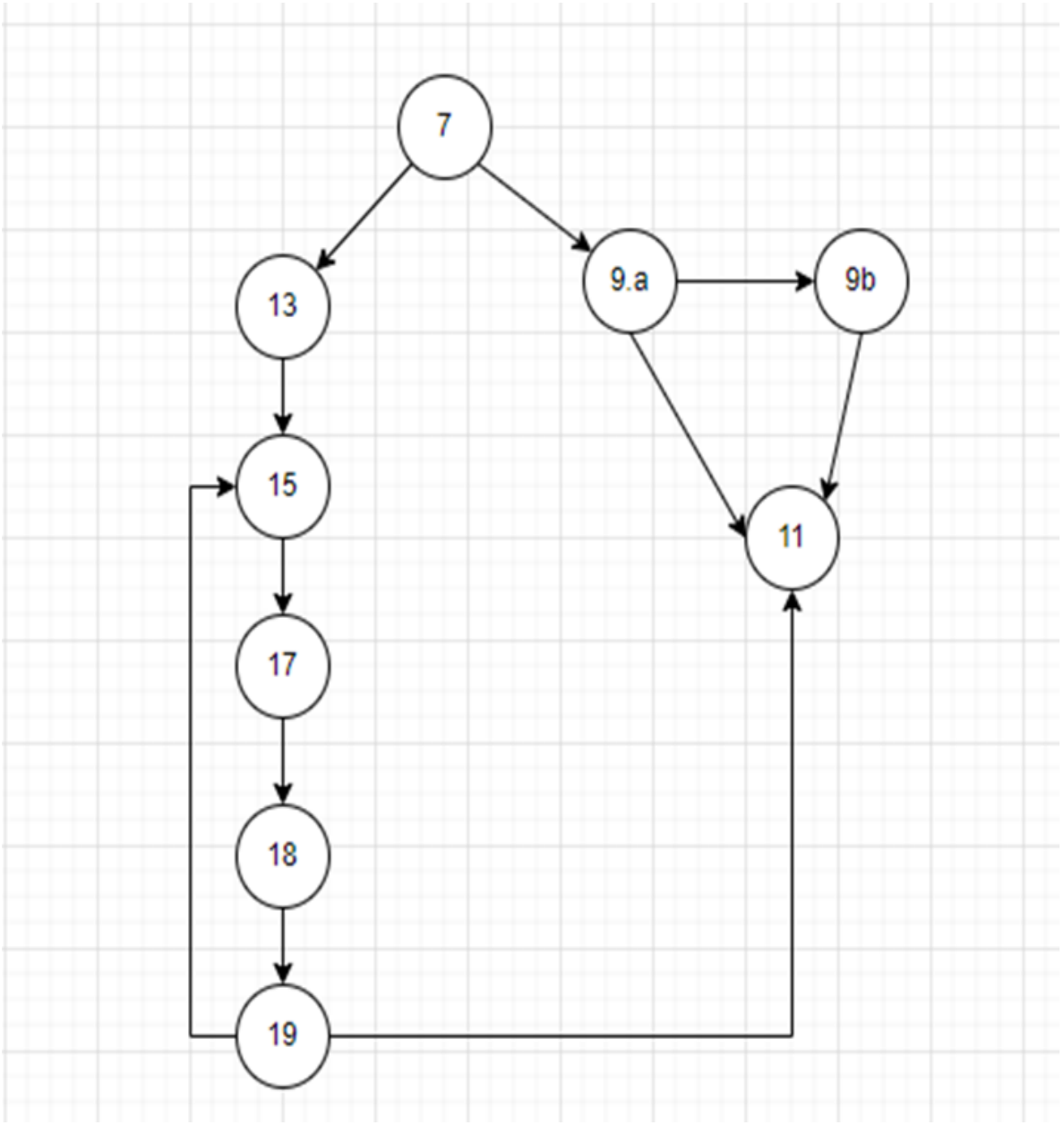
   a) Determinar las particiones de equivalencia para un programa, cuya especificación establece, que acepta de 4 a 8 entradas que son 5 dígitos enteros mayores que 10000.



b)

```
1   private static int fibonacci(int n)
2   {
3       int actual = 0;
4       int ant1, ant2;
5       ant1 = 1;
6       ant2 = 0;
7       if (n >= 0)
8       {
9           if ((n == 0) || (n == 1))
10          {
11              actual = n;
12          }
13          else
14          {
15              for (int i = 2; i <= n; i++)
16              {
17                  actual = ant1 + ant2;
18                  ant2 = ant1;
19                  ant1 = actual;
20              }
21          }
22      }
23      return actual;
24  }
```

PARTICIÓN	***EJEMPLO***	***RESULTADO ESPERADO***
0-1	0	0?
n>1	3	2
n<0	-1	0
n distinto int	‘Hola´	?
n> int. max	3.000.000	?



MÉTODOS:

1. V(G)=NP+1= 3
2. V(G)=A-N+2=10-9+2=3
3. V(G)=RC+1=3

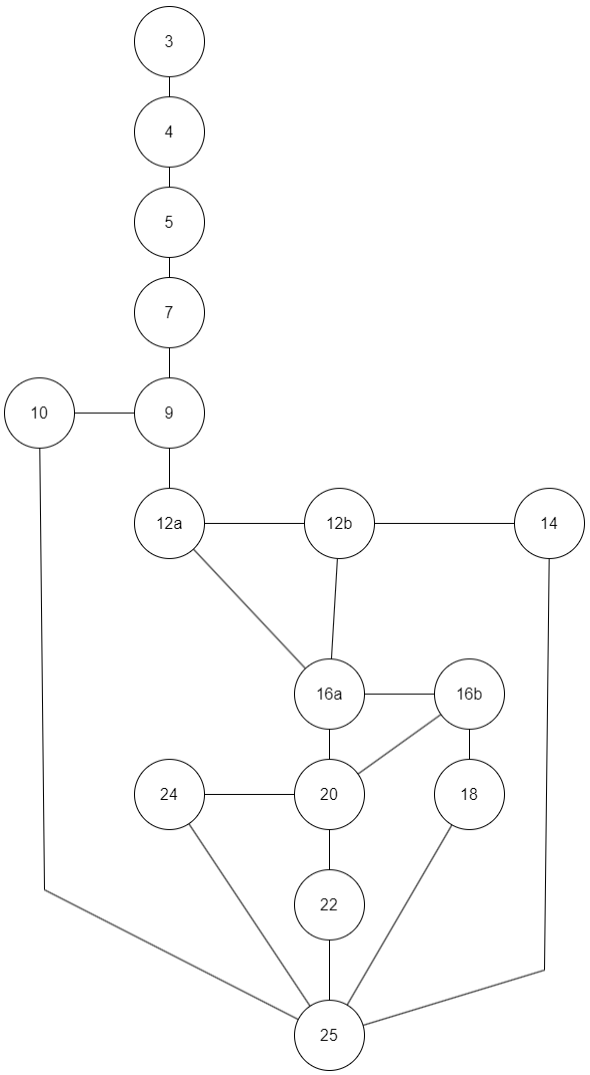
IDENTIFICACIÓN DE CAMINOS

- 7,9a,11
- 7,9a,9b,11
- 7,13,15,17,18,19,15
- 7,13,15,17,18,19,11

2. **Pruebas de caminos**

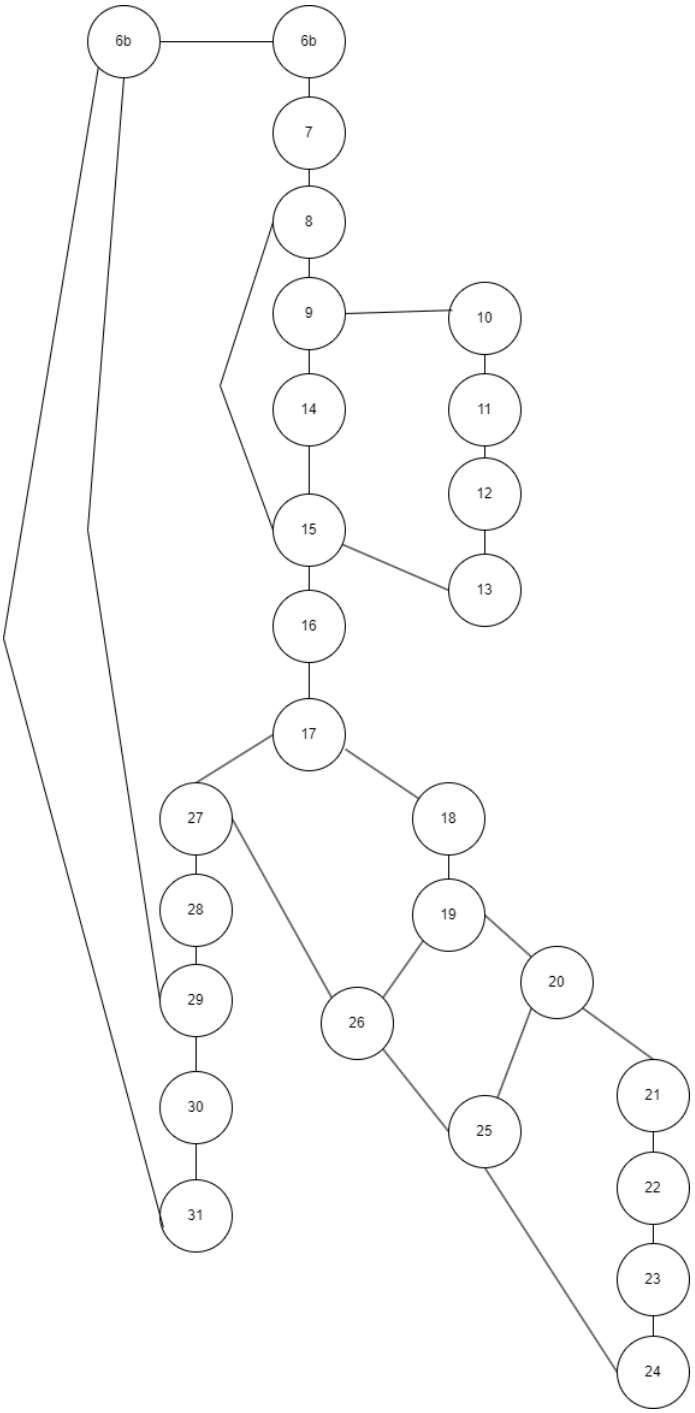
Realizar el grafo de flujo para el código del apartado b) del punto 1 y para los siguientes métodos. Calcular la complejidad ciclomática asociada por los tres métodos.

```
1   public class ReglaDeDescuento
2   {
3       private static final double porcentajeMenor = 0.03d;
4       private static final double porcentajeIntermedio = 0.05d;
5       private static final double porcentajeMayor = 0.10d;
6
7       public double Calcular(double total)
8       {
9           if(total <= 0)
10              throw new IllegalArgumentException("El total debe ser mayor a 0");
11
12          if (total > 5000 && total <= 10000)
13          {
14              return total * porcentajeMenor;
15          }
16          if (total > 10000 && total <= 25000)
17          {
18              return total * porcentajeIntermedio;
19          }
20          else if (total > 25000)
21          {
22              return total * porcentajeMayor;
23          }
24          return 0;
25      }
26  }
```



V(G)= NP +1 = 6+1= 7
V(G)=RC +1= 6+1= 7
V(G)= A-N+2= 21-16+2= 7

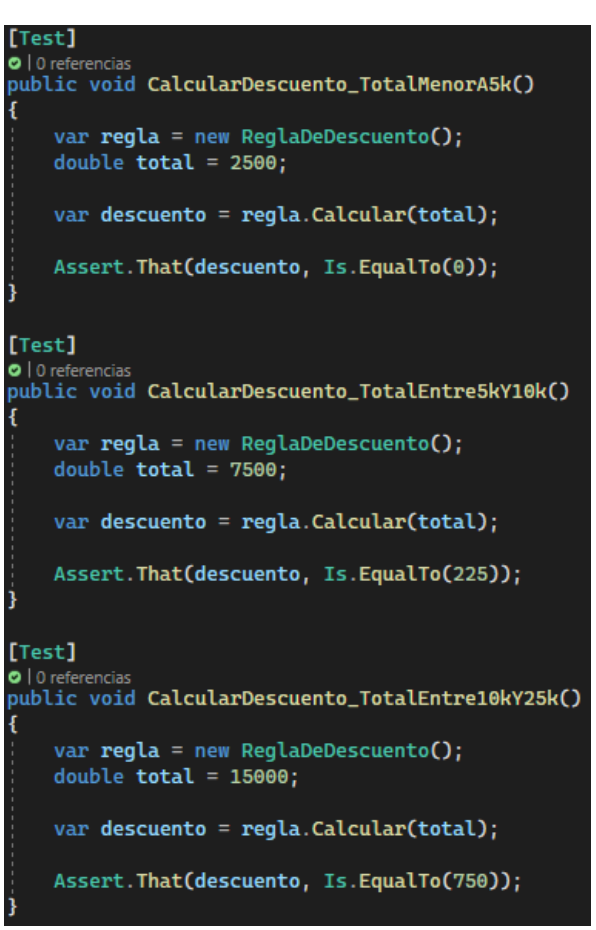
```java
public static int[] cocktailSort(int[] numbers)
{
    boolean swapped = true;
    int i = 0;
    int j = numbers.length - 1;
    while(i < j && swapped){
        swapped = false;
        for(int k = i; k < j; k++){
            if(numbers[k] > numbers[k + 1]){
                int temp = numbers[k];
                numbers[k] = numbers[k + 1];
                numbers[k + 1] = temp;
                swapped = true;
            }
        }
        j--;
        if(swapped){
            swapped = false;
            for(int k = j; k > i; k--){
                if(numbers[k] < numbers[k - 1]){
                    int temp = numbers[k];
                    numbers[k] = numbers[k - 1];
                    numbers[k - 1] = temp;
                    swapped = true;
                }
            }
        }
        i++;
    }
    return numbers;
}
```



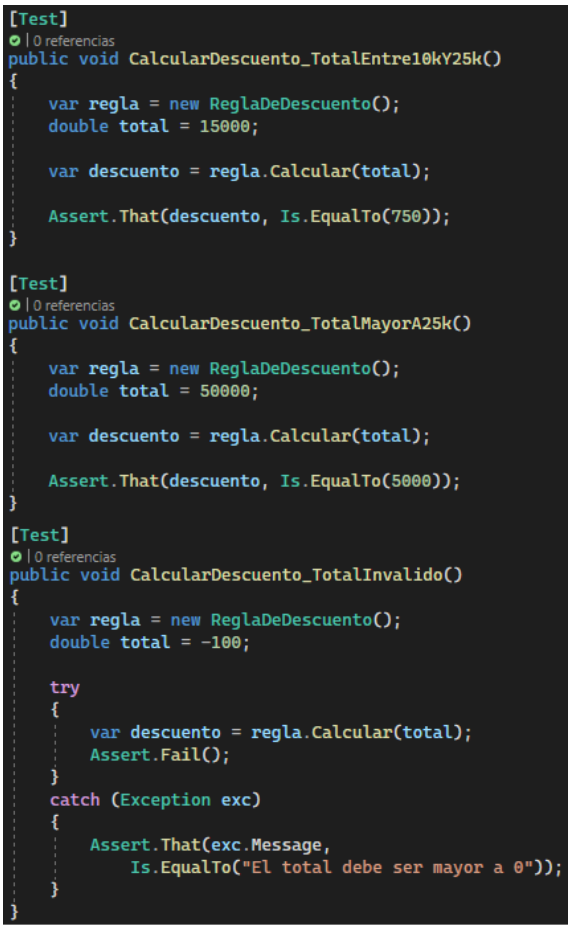
V(G)= NP +1 = 7+1= 8
V(G)=RC +1= 7+1= 8
V(G)= A-N+2= 35-29+2= 8

3. **Pruebas de Unidad (unitarias)**

Plantear las pruebas unitarias para la clase **ReglaDeDescuento**.



```
[Test]
0 referencias
public void CalcularDescuento_TotalMenorA5k()
{
    var regla = new ReglaDeDescuento();
    double total = 2500;

    var descuento = regla.Calcular(total);

    Assert.That(descuento, Is.EqualTo(0));
}

[Test]
0 referencias
public void CalcularDescuento_TotalEntre5kY10k()
{
    var regla = new ReglaDeDescuento();
    double total = 7500;

    var descuento = regla.Calcular(total);

    Assert.That(descuento, Is.EqualTo(225));
}

[Test]
0 referencias
public void CalcularDescuento_TotalEntre10kY25k()
{
    var regla = new ReglaDeDescuento();
    double total = 15000;

    var descuento = regla.Calcular(total);

    Assert.That(descuento, Is.EqualTo(750));
}
```



```
[Test]
0 referencias
public void CalcularDescuento_TotalEntre10kY25k()
{
    var regla = new ReglaDeDescuento();
    double total = 15000;

    var descuento = regla.Calcular(total);

    Assert.That(descuento, Is.EqualTo(750));
}

[Test]
0 referencias
public void CalcularDescuento_TotalMayorA25k()
{
    var regla = new ReglaDeDescuento();
    double total = 50000;

    var descuento = regla.Calcular(total);

    Assert.That(descuento, Is.EqualTo(5000));
}

[Test]
0 referencias
public void CalcularDescuento_TotalInvalido()
{
    var regla = new ReglaDeDescuento();
    double total = -100;

    try
    {
        var descuento = regla.Calcular(total);
        Assert.Fail();
    }
    catch (Exception exc)
    {
        Assert.That(exc.Message,
            Is.EqualTo("El total debe ser mayor a 0"));
    }
}
```

4. **Automatización de Pruebas de Aceptación y Pruebas Unitarias**

a) Automatizar, por lo menos, 3 (tres) escenarios en Gherkin realizados para el TP N° 2. b) Durante el proceso de automatización deberán realizarse, por lo menos, 3 (tres) pruebas unitarias.

5. **Pruebas de Versión (sistema)**

Para el caso de uso Realizar Venta diseñar 2 (dos) casos de prueba. Los casos se deben preparar en la plantilla que se adjunta.

Plantilla para caso de prueba

Caso de Prueba	
ID: 01	**Nombre: Crear una Nueva venta**
Descripción: Se creará una nueva venta	
Prioridad: Alta	**CU / HU: Realizar venta**
Módulo / Funcionalidad: Vendedor	
Diseñado por: Grupo 20	**Fecha: 15/11/23**
Ejecutado por: Grupo 20	**Fecha:15/11/23**
Precondiciones: **El vendedor debe estar autenticado.**	

Paso	Acción	Resultado Esperado	Pasó/ Fallo	Comentarios
1	Ingresar nueva venta		si	
2	Ingresar el código del producto 00256		si	
3	Seleccionar el color azul y talle L		si	
4	Seleccionar la forma de pago		si	
5	Confirmar la venta	Mensaje de creación exitosa. Registro almacenado correctamente en la base de datos.	si	

Caso de Prueba	
ID: 02	Nombre: Agregar Producto inexistente
Descripción: agregar producto inexistente	
Prioridad: Alta	CU / HU: Realizar venta
Módulo / Funcionalidad: Agregar Producto inexistente	
Diseñado por: Grupo 20	Fecha:15/11/23
Ejecutado por: Grupo 20	Fecha:15/11/23
Precondiciones: Usuario autenticado y con permisos para agregar productos. Existe una venta iniciada.	

Paso	Acción	Resultado Esperado	Pasó / Fallo	Comentarios
1	Ingresa el código del producto “P001”	Mensaje de error:”Producto inexistente”	si	
2				
3				
4				