UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL TUCUMÁN

Ingeniería de Software

TRABAJO PRÁCTICO N° 3
“Pruebas”

Grupo N°3

Integrantes:

[46246] Medina, Antonio Fabrizio
[49816] Abuin, Juan Jose
[42189] Ibañez Benjamin Martin
[42308] Reinoso Alvaro

Comisión 4K1

1) a) **Número de valores de entrada:**

Probar con valores frontera:

3 4 6 8 9

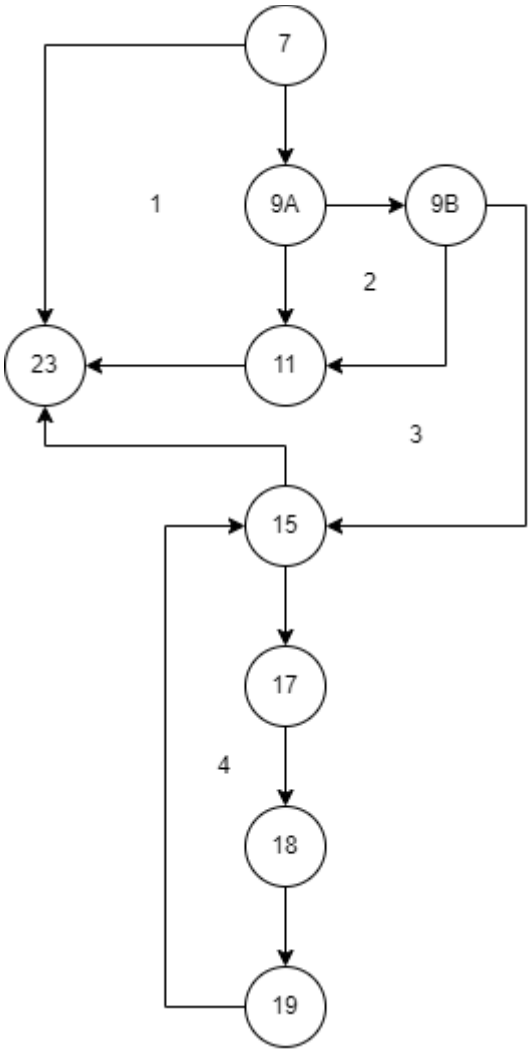
x < 4	4 ≤ x ≤ 8	x > 8

Valores de entrada:

9999 10000 50000 99999 100000

x < 10000	10000 ≤ x ≤ 99999	x > 99999

b) Para método **“Fibonacci”:**



Partición	Ejemplos	Resultado
n < 0	-1	0
n = 0 | n = 1	0 / 1	0 / 1
n > 1	3	2
n ≠ int	“ab”	?
n > int.max	3^100000	?
n < int.max	-3^100000	?

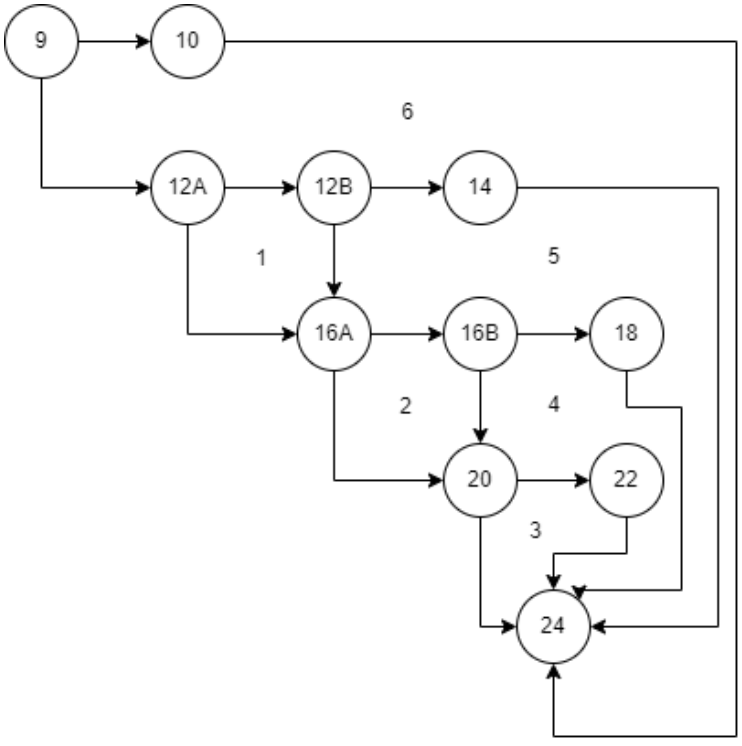
Complejidad Ciclomática:

- V(G) = Nodos predicados(4) + 1= **5**
- V(G) = Regiones encerradas(4) + 1 = **5**
- V(G) = Aristas(12) - Nodos(9) + 2 = **5**

Caminos:

- 7,9a,11, 23
- 7, 9a, 9b, 11, 23
- 7,23
- 7, 9a, 9b, 15, 23
- 7, 9a, 9b, 15, 15

2) a) Para método **“Calcular”:**



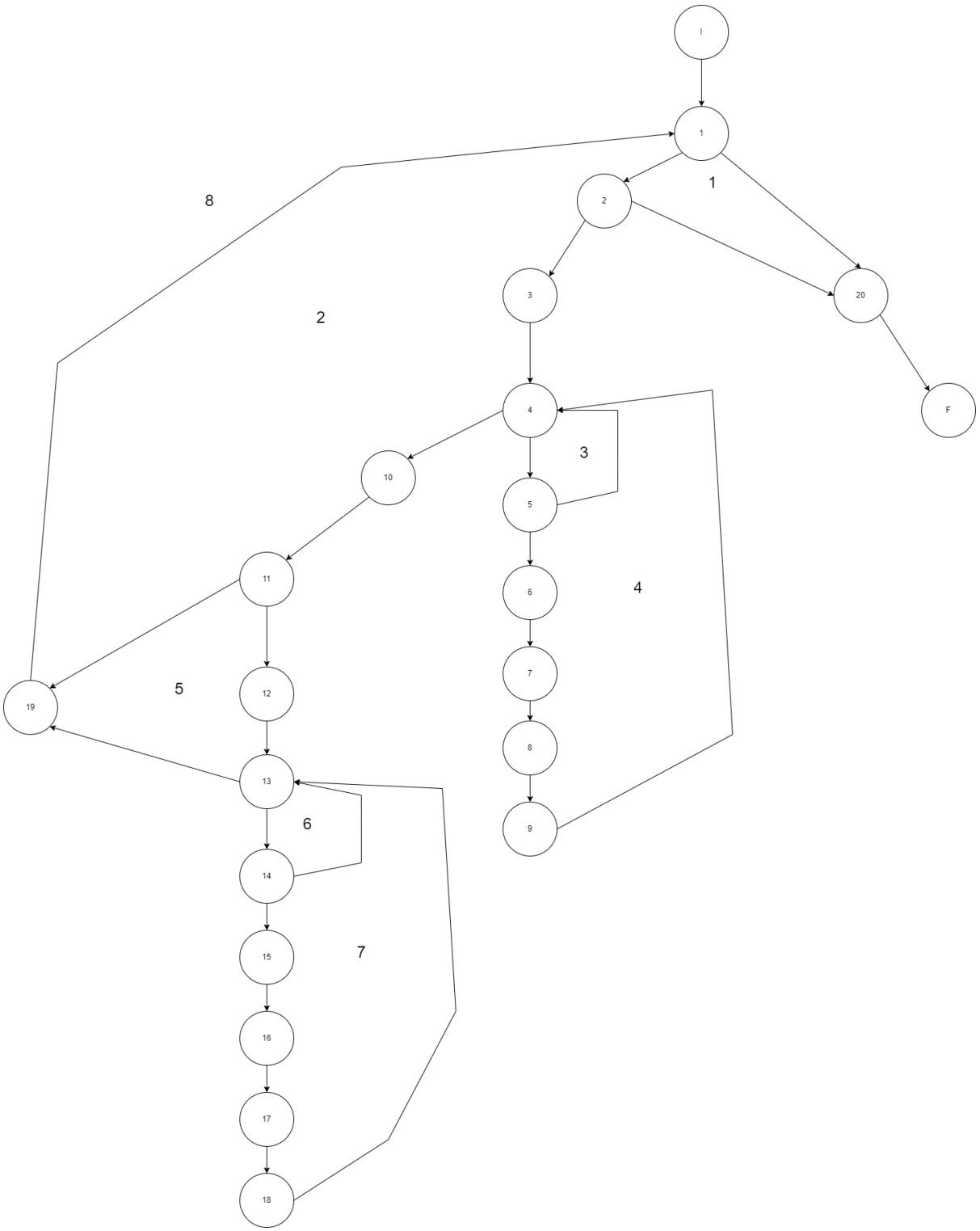
Complejidad Ciclomática:

- V(G) = Nodos predicados(6) + 1= **7**
- V(G) = Regiones encerradas(6) + 1 = **7**
- V(G) = Aristas(16) - Nodos(11) + 2 = **7**

Caminos:

- 9, 10, 24
- 9, 12A, 12B, 14, 24
- 9, 12A, 16A, 16B, 18, 24
- 9, 12A, 12B, 16A, 16B, 18, 24
- 9, 12A, 12B, 16A, 20, 24
- 9, 12A, 12B, 16A, 20, 22, 24
- 9, 12A, 16A, 20, 24

b) Para método **“Cocktail sort”:**



Complejidad Ciclomática:

- V(G) = Nodos predicados(7) + 1= **8**
- V(G) = Regiones encerradas(8) = **8**
- V(G) = Aristas(28) - Nodos(22) + 2 = **8**

5)

Caso de Prueba			
ID:01	**Nombre:Realizar pago con tarjeta de crédito/débito**		
Descripción:Realizar un nuevo pago con tarjeta de crédito/débito			
Prioridad:alta		**CU / HU:Realizar venta**	
Módulo / Funcionalidad: Pago			
Diseñado por:			**Fecha:**
Ejecutado por:			**Fecha:**

Precondiciones: Hay una venta en curso

Paso	**Acción**	**Resultado Esperado**	**Pasó / Falló**	**Comentarios**
1	**El cliente informa al vendedor que desea abonar con tarjeta.**	**Verificar que el sistema está preparado para procesar el pago con tarjeta y que el monto es correcto.**		
2	**El vendedor ingresa los datos de la tarjeta**	**El sistema acepta los detalles de la tarjeta y muestra un resumen del pago**		**Verificar que el sistema acepta los detalles de la tarjeta y que el resumen de pago sea preciso.**
3	**El vendedor confirma**	**Se muestra un mensaje de confirmación y se procesa el pago con éxito.**		
4	**El vendedor emite el comprobante de compra.**	**Verificar que el comprobante se emita correctamente y contenga la información relevante.**		

Caso de Prueba			
ID:02	**Nombre: Realizar pago con efectivo**		
Descripción: Realizar un nuevo pago con efectivo			
Prioridad:alta		**CU / HU: Realizar venta**	
Módulo / Funcionalidad: Pago			
Diseñado por:			**Fecha:**
Ejecutado por:			**Fecha:**

Precondiciones: Existe una venta en curso. **Los datos del cliente están asociados a la venta.**

Paso	Acción	Resultado Esperado	Pasó / Falló	Comentarios
1	Informar al vendedor que se hará un pago en efectivo	El vendedor inicia el proceso de pago en efectivo en el sistema de punto de venta		
2	El cliente entrega el monto exacto de efectivo	El vendedor recibe el pago en efectivo y verifica que el monto sea correcto		
3	El vendedor registra el pago en efectivo	El sistema registra correctamente el pago y actualiza el stock.		
4	El vendedor emite el comprobante de pago.	Verificar que el comprobante se emita correctamente y contenga la información relevante.		

3)

```
	@Test
	public void
totalEntre5001y10000aplicaPorcentajeMenorDeDescuento(){
		public double descuento;
		public double esperado = 180.0;

		public ReglaDeDescuento regla = new ReglaDeDescuento();

		descuento = regla.Calcular(6000.0);

		assertEquals(esperado, descuento);
	}


	@Test
	public void
totalEntre10001y25000aplicaPorcentajeIntermedioDeDescuento(){
		public double descuento;
		public double esperado = 1000.0;

		public ReglaDeDescuento regla = new ReglaDeDescuento();

		descuento = regla.Calcular(20000.0);

		assertEquals(esperado, descuento);
	}

	@Test
	public void
totalMayorA25000AplicaPorcentajeMayorDeDescuento(){
		public double descuento;
		public double esperado = 3000.0;

		public ReglaDeDescuento regla = new ReglaDeDescuento();

		descuento = regla.Calcular(30000.0);

		assertEquals(esperado, descuento);
	}
```

```
@Test
public void totalEntre1y5000noAplicaDescuento(){
      public double descuento;
      public double esperado = 0.0;

      public ReglaDeDescuento regla = new ReglaDeDescuento();

      descuento = regla.Calcular(5000.0);

      assertEquals(esperado, descuento);
}


@Test
public void totalMenorOIgualA0LanzaExepcion(){
      public ReglaDeDescuento regla = new ReglaDeDescuento();

      assertThrows(IllegalArgumentException.class, ()->{
            regla.Calcular(0.0)
      });
}
```