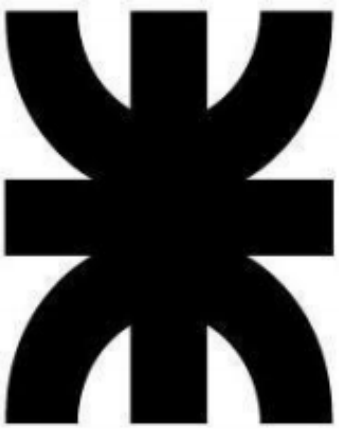
TRABAJO PRACTICO NRO 3



Catedra de Ing. de Software

Docente: Ing. Vicente, Francisco

Fecha de Entrega: 14 de noviembre de 2023

Grupo 2

ALUMNOS:

- LLEBEILI, DANIEL AGUSTÍN 46.327
- RODRIGUEZ, MARIANO NICOLAS 46.351
- SICHI, FACUNDO 48.124
- VIGIANI, RICARDO NICOLÁS 48.153

COMISION: 4K1

PUNTO 1 - PRUEBA 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.

Particiones	**Entrada**	**Salida**
Números menores a 10000	9000	Invalida
Numero exactamente 10000	10000	Invalida
Números entre 10000 y 99999	50000	Valida
Números mayores a 99999	100000	Invalida
Entradas menores a 4	3 entradas	Invalida
Entradas entre 4 y 8	5 entradas	Valida
Entradas mayores a 8	9 entradas	Invalida

B)

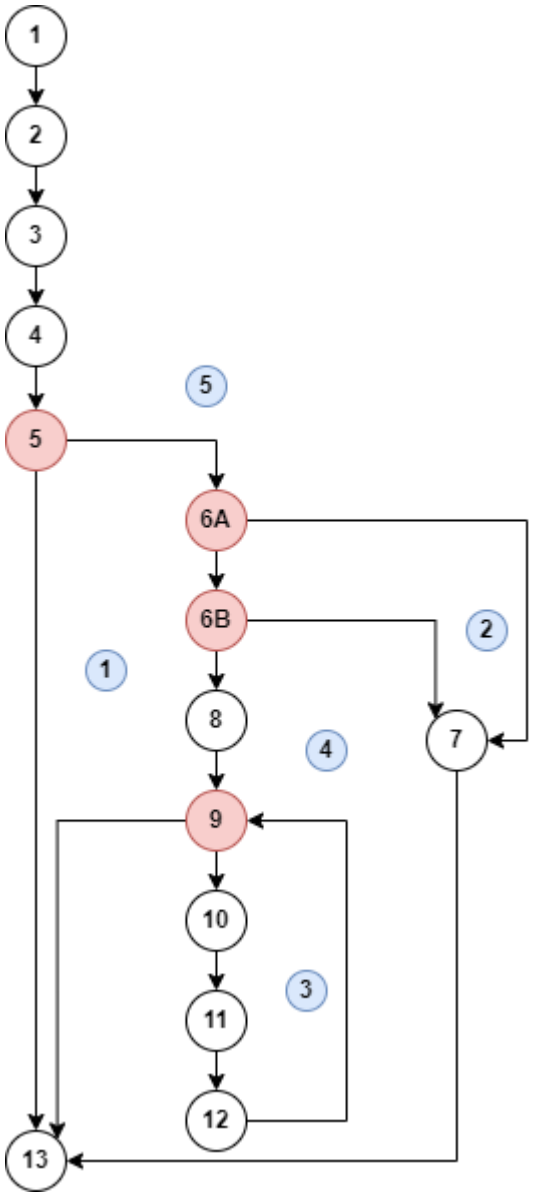
```
1   private static int fibonacci(int n)
2   {
3       int actual = 0;
4       int ant1, ant2;
5       ant1 = 1;
6       ant2 = 0;
7       if (n >= 0)
8       {
9           if ((n == 0) || (n == 1))
10          {
11              actual = n;
12          }
13          else
14          {
15              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  }
```

Particiones	**Entrada**	**Salida**
Números no tipo Integer	- 2.147.483.648 2.147.483.648 3,498	ERROR
- 2.147.483.647 ≤ n < 0	-1	0
n = 0	0	0
n = 1	1	1
1< n ≤ 2.147.483.647	3	2

PUNTO 2 - PRUEBA DE CAMINOS

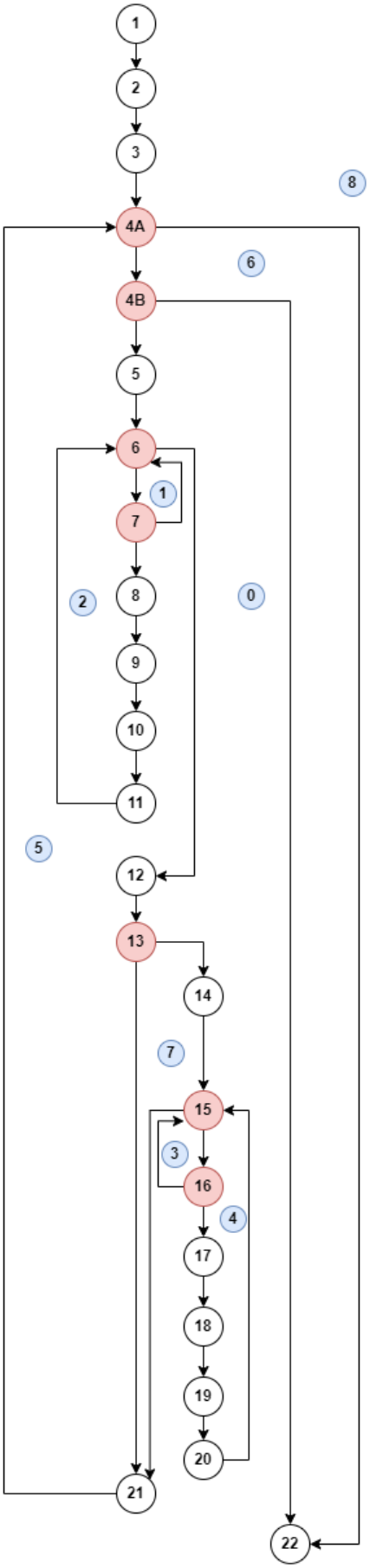
Realizar el grafo de flujo para el código del apartado b) del punto 1 y para los siguientes métodos.

Calcular la complejidad celomática asociada por los tres métodos.




```
1. int actual = 0;
2. int ant1, ant2;
3. ant1 = 1;
4. ant2 = 0;
5. if(n > 0)
6.    if ((n == 0) || (n ==1 ))
7.    actual = n;
8.    else
9.    for(int i = 2; i <= n; i++)
10.    actual = ant1 + ant2;
11.    ant2 = ant1;
12.    ant1 = actual;
13. return actual;
```

V(G) = Aristas – nodos + 2 V(G) = 17 - 14 + 2 V(G) = 5	V(G) = Nodos predicado V(G) = 4 + 1 V(G) = 5	V(G) = Regiones V(G) = 5




```
1.boolean swapped =true;
2.int i = 0;
3.int j = numbers.lenght -1;
4.while (i<j && swapped)
5       swapped = false;
6.for (int k= i; k<j;k++)
7.if(numbers[k]>numbers[k+1])
8.      int temp = numbers [k];
9.      numbers[k] = numbers[k+1];
10.      numbers[k+1] = temp;
11.      swapped = true;
12.j--;
13. if(swapped)
14.swapped = false;
15.for(int k = j;k>i;k--)
16.   if(numbers[k]<numbers[k-1])
17.      int temp=numbers[k];
18.      numbers[k]=numbers[k-1];
19.      numbers[k-1]=temp;
20.      swapped = true;
21.i++
22.return numbers
```

V(G) = Aristas – nodos + 2 V(G) = 29 - 23 + 2 V(G) = 8	V(G) = Nodos predicado V(G) = 7 + 1 V(G) = 8	V(G) = Regiones V(G) = 8

PUNTO 3 - PRUEBAS DE UNIDAD (UNITARIAS)

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

```java
@Test
    public void calcularDescuentoParaMontoIgualA0() {
        ReglaDeDescuento reglaDeDescuento = new ReglaDeDescuento();
        double total = 0;
        double descuentoEsperado = 0;
        try{
            double descuentoActual = reglaDeDescuento.Calcular(total);
            fail();
        }catch(IllegalArgumentException error){
            assertEquals(error.getMessage(), "El total debe ser mayor a 0");
        }
    }
```

```java
@Test
public void calcularDescuentoParaMontoEntre5000Y10000() {
    ReglaDeDescuento reglaDeDescuento = new ReglaDeDescuento();
    double total = 6000;
    double descuentoEsperado = total * 0.03d;
    double descuentoActual = reglaDeDescuento.Calcular(total);
    assertEquals(descuentoEsperado, descuentoActual);
}
```

```java
@Test
public void calcularDescuentoParaMontoIgualA5000() {
    ReglaDeDescuento reglaDeDescuento = new ReglaDeDescuento();
    double total = 5000;
    double descuentoEsperado = 0;
    double descuentoActual = reglaDeDescuento.Calcular(total);
    assertEquals(descuentoEsperado, descuentoActual);
}
```

```java
@Test
public void calcularDescuentoParaMontoIgualA10000() {
    ReglaDeDescuento reglaDeDescuento = new ReglaDeDescuento();
    double total = 10000;
    double descuentoEsperado = total * 0.03d;
    double descuentoActual = reglaDeDescuento.Calcular(total);
    assertEquals(descuentoEsperado, descuentoActual);
}
```

```java
@Test
public void calcularDescuentoParaMontoEntre10000Y25000() {
    ReglaDeDescuento reglaDeDescuento = new ReglaDeDescuento();
    double total = 15000;
    double descuentoEsperado = total * 0.05d;
    double descuentoActual = reglaDeDescuento.Calcular(total);
    assertEquals(descuentoEsperado, descuentoActual);
}
```

```java
@Test
public void calcularDescuentoParaMontoIgualA25000() {
    ReglaDeDescuento reglaDeDescuento = new ReglaDeDescuento();
    double total = 25000;
    double descuentoEsperado = total * 0.05d;
    double descuentoActual = reglaDeDescuento.Calcular(total);
    assertEquals(descuentoEsperado, descuentoActual);
}
```

```java
@Test
public void calcularDescuentoParaMontoMayorA25000() {
    ReglaDeDescuento reglaDeDescuento = new ReglaDeDescuento();
    double total = 30000;
    double descuentoEsperado = total * 0.10d;
    double descuentoActual = reglaDeDescuento.Calcular(total);
    assertEquals(descuentoEsperado, descuentoActual);
}
```

PUNTO 4 – AUTOMATIZACION DE PRUEBAS DE ACEPTACION 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.

Nota: se adjunta resolución de este apartado en el repositorio correspondiente.

PUNTO 5 – PRUEBAS DE VERSION (SISTEMA)

Caso de Prueba		
ID: 01	**Nombre:** Registrar nueva venta	
Descripción: Crear una nueva venta con una única línea de venta.		
Prioridad: Alta	**CU / HU:** Realizar venta	
Módulo / Funcionalidad: Ventas		
Diseñado por: Grupo 2		**Fecha:** 12/11
Ejecutado por: -		**Fecha:** 12/11

Precondiciones:

- Vendedor autenticado y verificado
- Artículos existentes en inventario
- Talles y colores disponibles.

Paso	Acción	Resultado Esperado	Pasó / Falló	Comentarios
1	Presionar el botón de iniciar una nueva venta			
2	Ingresar el código del articulo	Se muestra la información del articulo y combinaciones disponibles.		
3	Seleccionar el talle, color y cantidad (*)			
4	Confirmar la combinación seleccionada para ese articulo	Se muestra el artículo en una línea de venta dentro de la venta. Se visualiza el subtotal y total actualizados. (Mayor a $92700)		
5	Ingresar el DNI del cliente.	Se visualiza la información de la venta actualizada.		
6	Seleccionar el método de pago Efectivo	Se previsualiza el tipo de comprobante a emitir con los detalles de la venta		
7	Presionar el botón de validar comprobante	Se emite el comprobante.		
8	Finalizar Venta	Mensaje de creación exitosa. Registro almacenado en la base de datos.		