UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL TUCUMÁN

INGENIERÍA DEL SOFTWARE

TRABAJO PRÁCTICO N° 3

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

2023

4k2

Grupo

10

Alicata Matías Jesús 42135

Diaz Daniela Rocio 48223

Negro Luqui Franco 53942

Herrera Macarena Del Valle

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.

Invalido	Valido	Invalido
Numero de entradas < 4	Numero de entradas entre [4-8]	Numero de entradas > 8
Valores < 10000	Valores entre [10000 - 99999]	Valores > 99999

b)

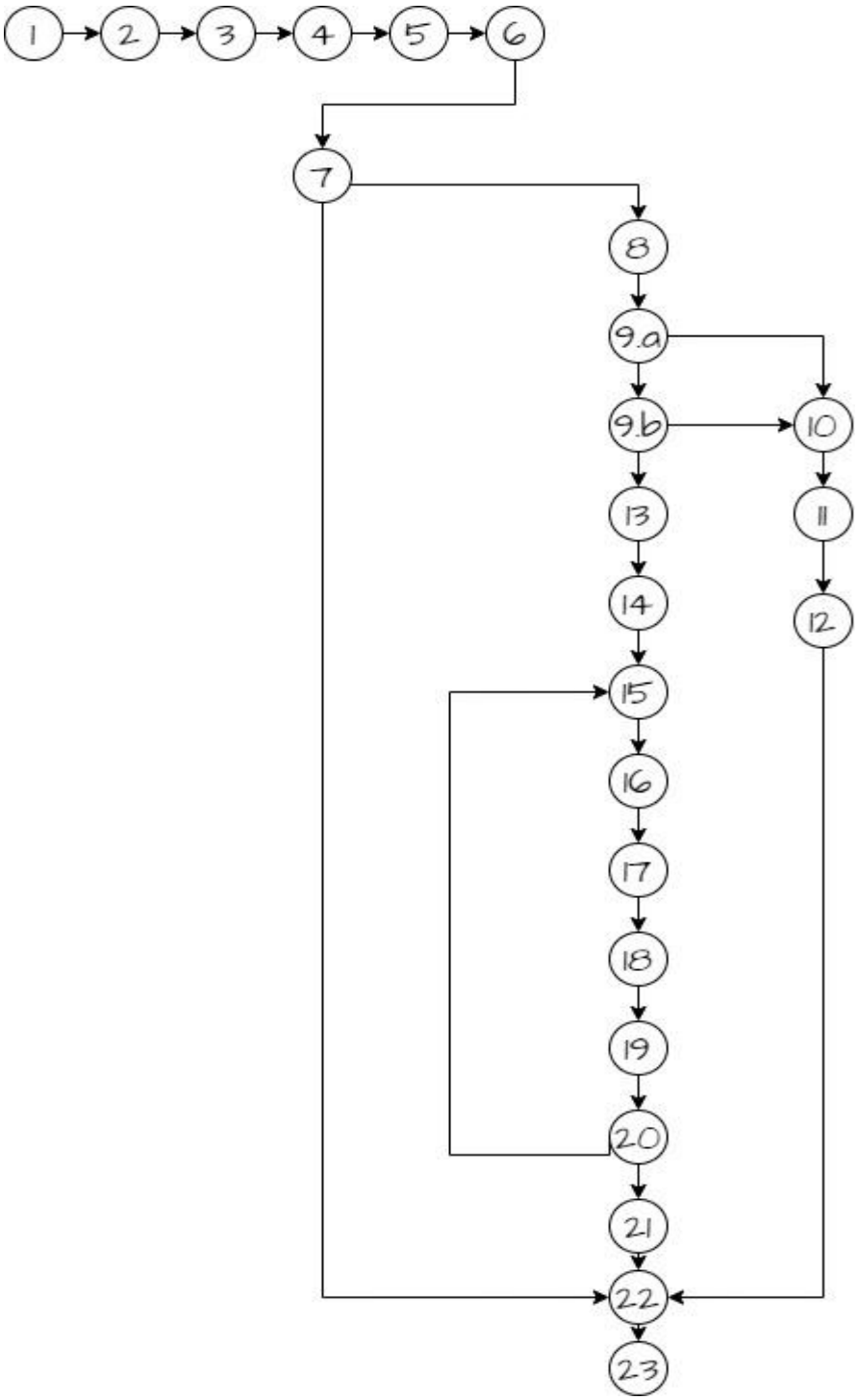
```
1   private static int fibonacci(int n)
2   {
3       int actual = 0;
4       int ant1, ant2;
5       ant1 = 1;
6       ant2 = 0;
7       if (n >= 0)
8       {
9           if ((n == 0) || (n == 1))
10          {
11              actual = n;
12          }
13          else
14          {
15         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

Numero de particion	Particion
1	n < 0
2	n > 1
3	n = [0-1]
4	n != int

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.

Apartado b punto 1:

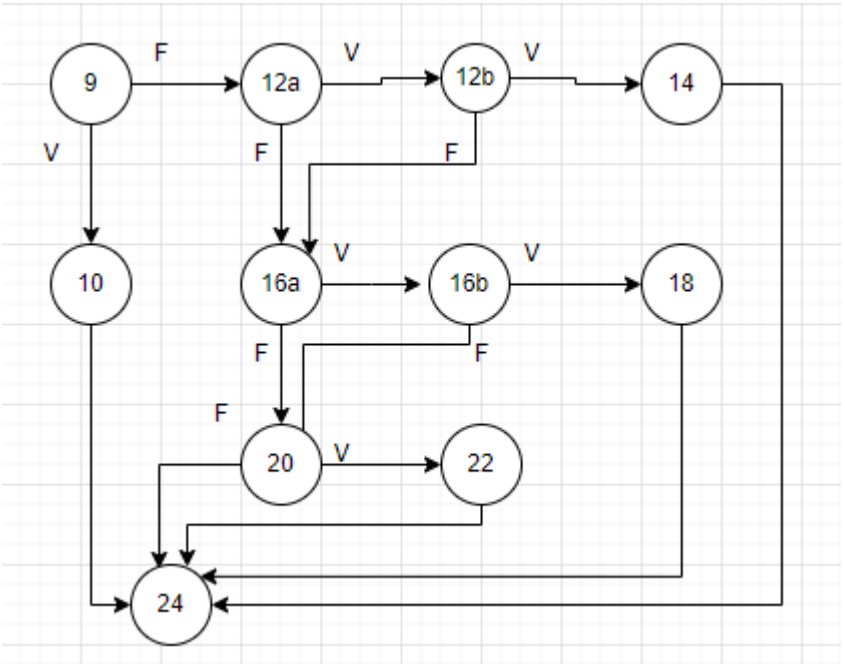


CÁLCULO DE LA COMPLEJIDAD CICLOMÁTICA

$V(G) = cantidad\ de\ aristas\ -\ cantidad\ de\ nodos\ +\ 2$
$V(G) = 27\ -\ 24\ +\ 2 \rightarrow V(G) = 5$
$V(G) = cantidad\ de\ nodos\ predicados\ +\ 1$
$V(G) = 4\ +\ 1 \rightarrow V(G) = 5$
$V(G) = cantidad\ de\ regiones \rightarrow V(G) = 5$

Regla de Descuento

```
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  }
```

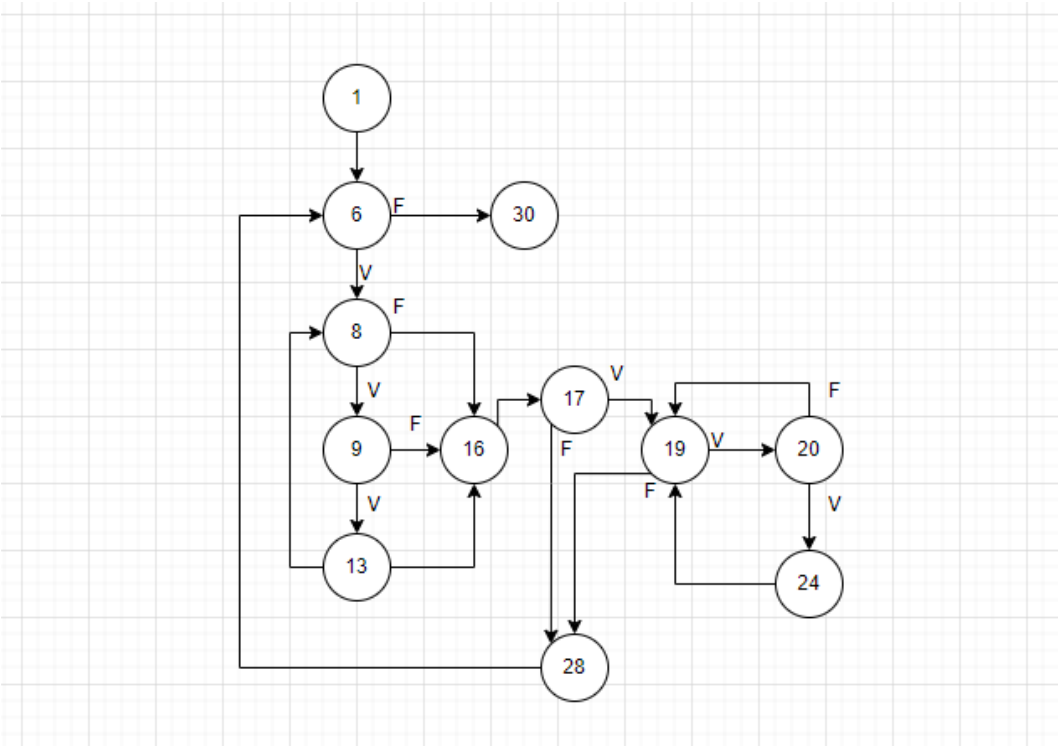


CÁLCULO DE LA COMPLEJIDAD CICLOMÁTICA

$V(G) = cantidad\ de\ aristas\ -\ cantidad\ de\ nodos\ +\ 2$
$V(G) = 16\ -\ 11\ +\ 2 \rightarrow V(G) = 7$
$V(G) = cantidad\ de\ nodos\ predicados\ +\ 1$
$V(G) = 6\ +\ 1 \rightarrow V(G) = 7$
$V(G) = cantidad\ de\ regiones \rightarrow V(G) = 7$

C.

```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;
}
```



CÁLCULO DE LA COMPLEJIDAD CICLOMÁTICA

$V(G) = cantidad\ de\ aristas\ -\ cantidad\ de\ nodos\ +\ 2$
$V(G) = 12\ -\ 10\ +\ 2 \rightarrow V(G) = 5$
$V(G) = cantidad\ de\ nodos\ predicados\ +\ 1$
$V(G) = 4\ +\ 1 \rightarrow V(G) = 5$
$V(G) = cantidad\ de\ regiones \rightarrow V(G) = 5$

3. Pruebas de Unidad (unitarias)

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

```
@Test
public void calcularDescuentoConTotalMenora0yTiraExcepcion() {
    var desc = new ReglaDeDescuento()
    try {
        desc.Calcular(-5);
        fail();
    } catch (Exception error){
        assertEquals("El total debe ser mayor a 0", error.getMessage());
    }
}
```

```
@Test
public void calcularDescuentoConTotalIgualA0yTiraExcepcion() {
    var desc = new ReglaDeDescuento()
    try {
        desc.Calcular(0);
        fail();
    } catch (Exception error){
        assertEquals("El total no puede ser igual a 0", error.getMessage());
    }
}
```

```
@Test
public void entre5000y10000elPorcentajeDebeSerMenor(){
    var descMenor = new ReglaDeDescuento();
    valor = descMenor.Calcular(7000);
    assertEquals(valor, 210);
}
```

```java
@Test
public void entre10000y25000elPorcentajeDebeSerIntermedio(){
    var descIntermedio = new ReglaDeDescuento();
    valor = descIntermedio.Calcular(15000);
    assertEquals(valor, 750);

}

@Test
public void totalMayorA25000ElPorcentajeDebeSerMayor(){
    var descMayor = new ReglaDeDescuento();
    valor = descMayor.Calcular(50000);
    assertEquals(valor, 5000);
}
```

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.

https://github.com/ing-software-frt-utn/tp3-2023/tree/4k2-G10

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.

Caso de Prueba		
ID: 01	**Nombre: Eliminar Artículo de la venta**	
Descripción: Eliminar un artículo de la venta.		
Prioridad: Media/Alta	**CU / HU: Realizar Venta**	
Módulo / Funcionalidad: Ventas		
Diseñado por: Grupo 10		**Fecha: 12/12/23**
Ejecutado por: -		**Fecha: 12/12/23**

Precondiciones: Venta en proceso **Uno o más artículos seleccionados**

Paso	Acción	Resultado Esperado	Pasó / Falló	Comentarios
1	Seleccionar el artículo deseado para eliminar.	Se selecciona el artículo en la tabla		
2	Confirmar la eliminación	Se elimina el artículo en la tabla		
3		Se actualiza el subtotal de la venta		
4				

ID: Identificador | CU: Caso de Uso | HU: Historia de Usuario

Caso De Prueba		
ID: 02	**Nombre:Registrar nueva venta flujo básico**	
Descripción: El vendedor desea registrar una venta con productos cargados		
Prioridad: Alta	**CU/HU: Realizar venta**	
Módulo/Funcionalidad: Ventas		
Diseñado por: Grupo 10		**Fecha: 17/11**
Ejecutado por: -		**Fecha: 17/11**

Precondiciones: **Vendedor autenticado y verificado** **Artículos cargados en la venta** **Talles y colores disponibles**

Paso	Acción	Resultado Esperado	Pasó /Falló	Comentarios
1	Pulsar el botón de iniciar una nueva venta	El cliente es consumidor final		

2	Se ingresa el código del producto	Se muestra el producto		
3	Se selecciona el talle, color y cantidad	Se visualiza la actualización del producto		
4	Se confirma el producto	Se agrega el producto a la venta y se actualiza el total		
5	Se elige el tipo de factura	Se visualiza el tipo de factura		
6	Se confirma la venta	Mensaje de creación exitosa y almacenado automáticamente.		

Nota: el trabajo será entregado a través de un repositorio Git a definir.