Ejercicio - 1

Código:

```
1 if (a > 1 and b > 5 and c < 2) then
2 x = x + 1
3 else
4 x = x - 1
5 end
```

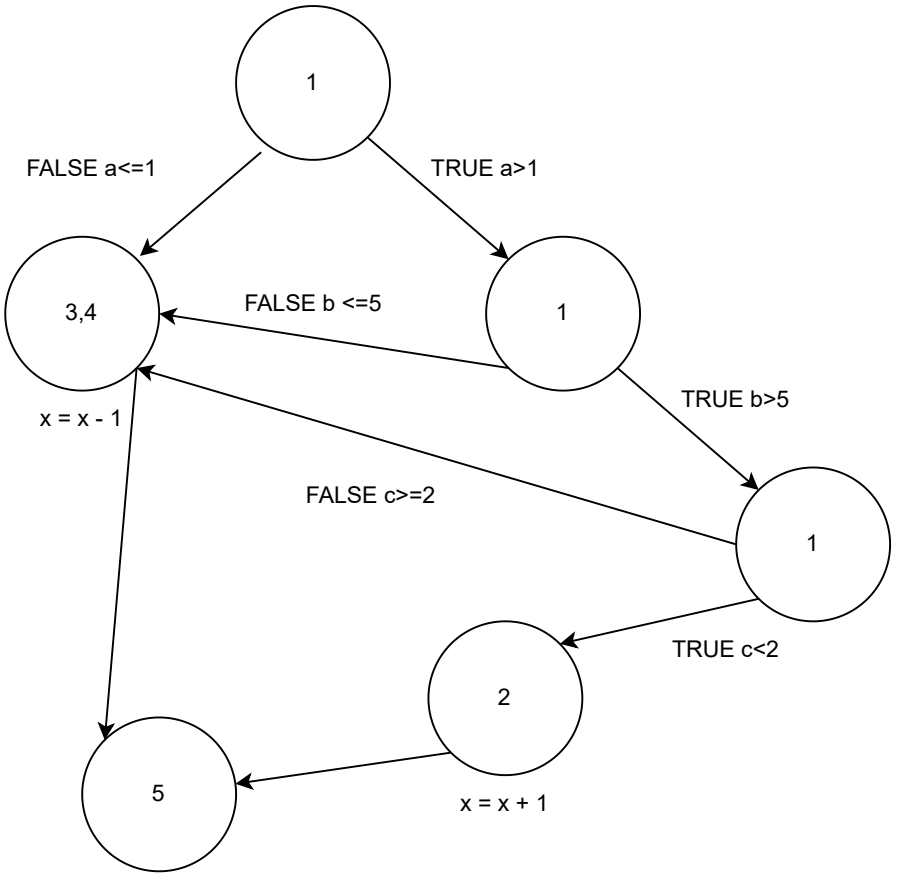
Complejidad Ciclomática:

V(G) = A-N+2 = 4, Hay 8 Aristas y 6 Nodos

ó

V(G) = nps + 1 = 4, Hay 3 Nodos con Dos Aristas

Programa Simple Sin Mucho Riesgo.



Ejercicio - 2

Código:

```
1 if (a > 1 or b > 5 or c < 2) then
2 x = x + 1
3 else
4 x = x - 1
5 end
```

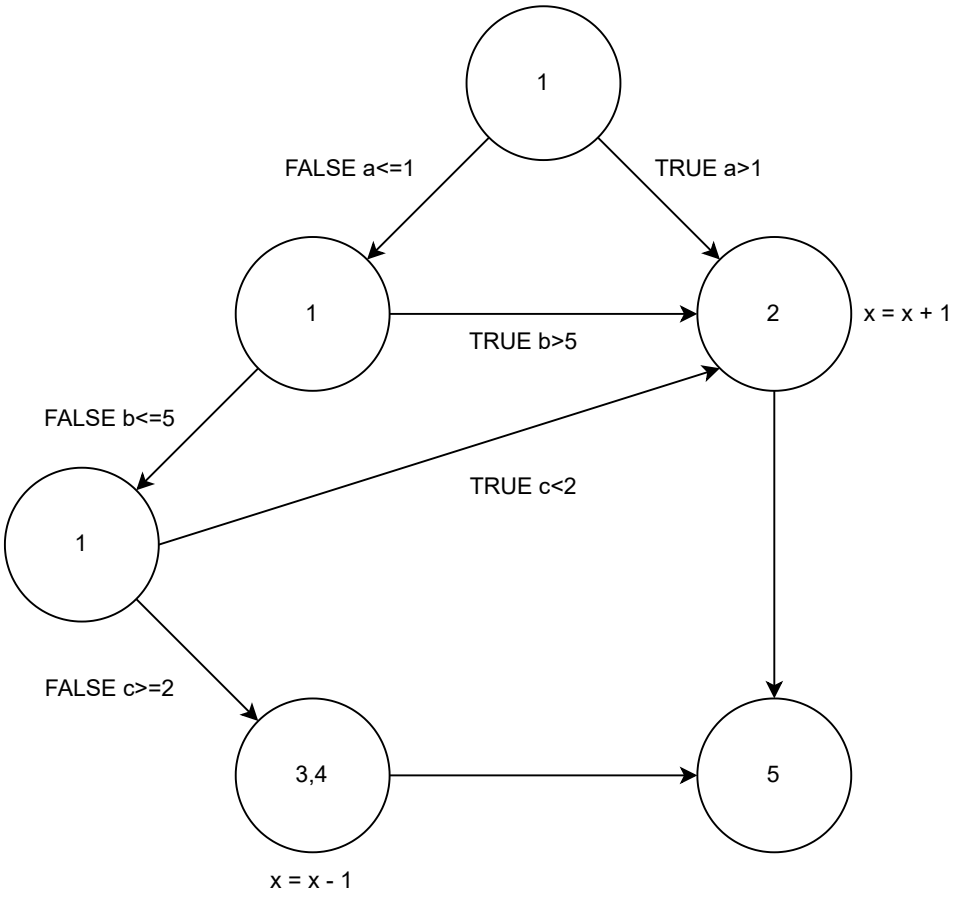
Complejidad Ciclomática:

V(G) = A - N + 2 = 4 Hay 8 Aristas y 6 Nodos.

ó

V(G) = nps + 1 = 4 Hay 3 Nodos con 2 Aristas.

Programa Simple sin Mucho Riesgo.



Ejercicio - 3

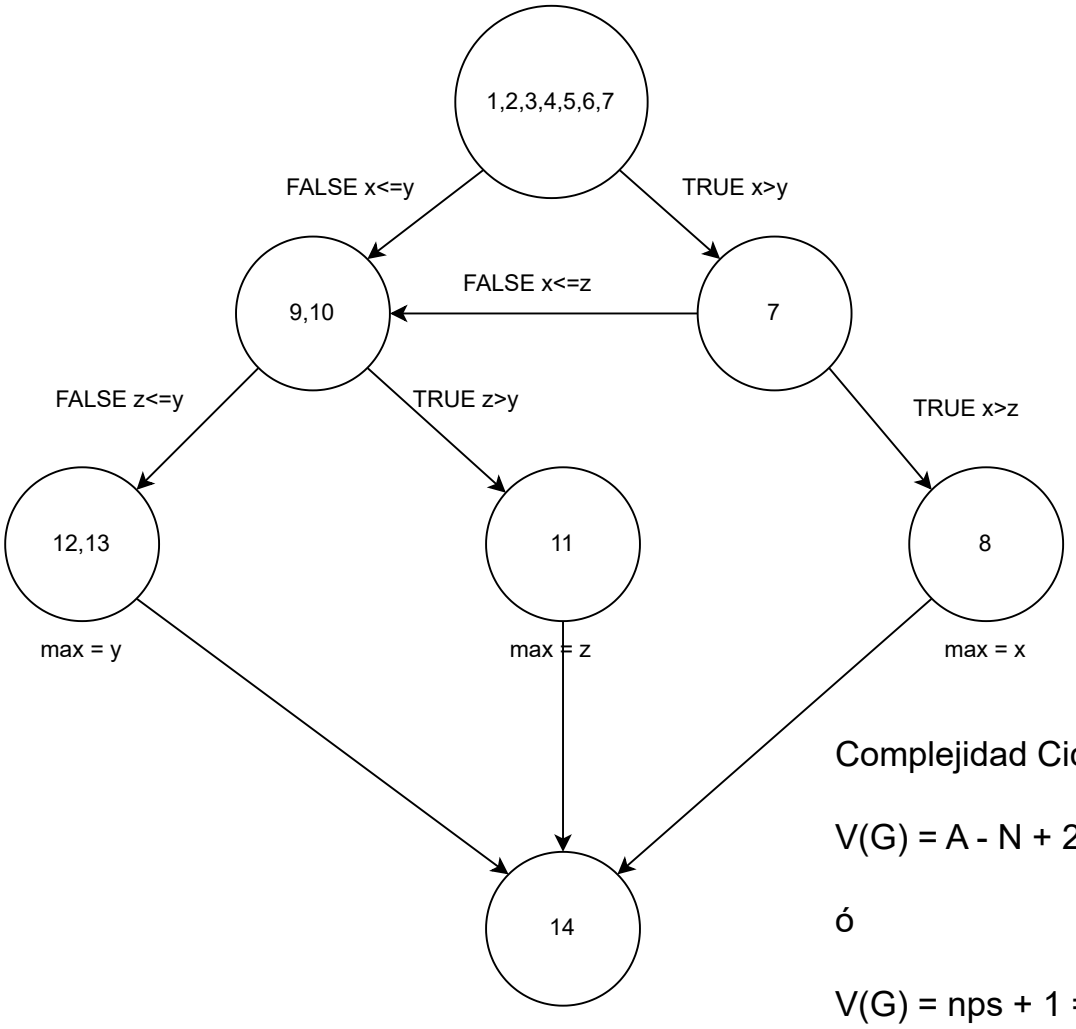
```
Import java.io.*;

Public class Maximo
{
        public static void main (String args[]) throws IOException
        {
1       BufferedReader entrada = new BufferedReader (new InputStreamReader (System.in));

2       int x,y,z,max;

3       System.out.println("Introduce x,y,z: ");
4       x =  Integer.parseInt (entrada.readLine());
5       y =  Integer.parseInt (entrada.readLine());
6       z =  Integer.parseInt (entrada.readLine());

7       if (x>y && x>z)
8               max = x;
9       else{
10              if (z>y)
11                      max = z;
12              else
13                      max = y;
        }
14      System.out.println ("El máximo es "+ max);
        } //main
}
```



Muestra El Número de Mayor Valor.

Complejidad Ciclomática:

V(G) = A - N + 2 = 4, Hay 9 Aristas y 7 Nodos.

ó

V(G) = nps + 1 = 4, Hay 3 Nodos con 2 Aristas.

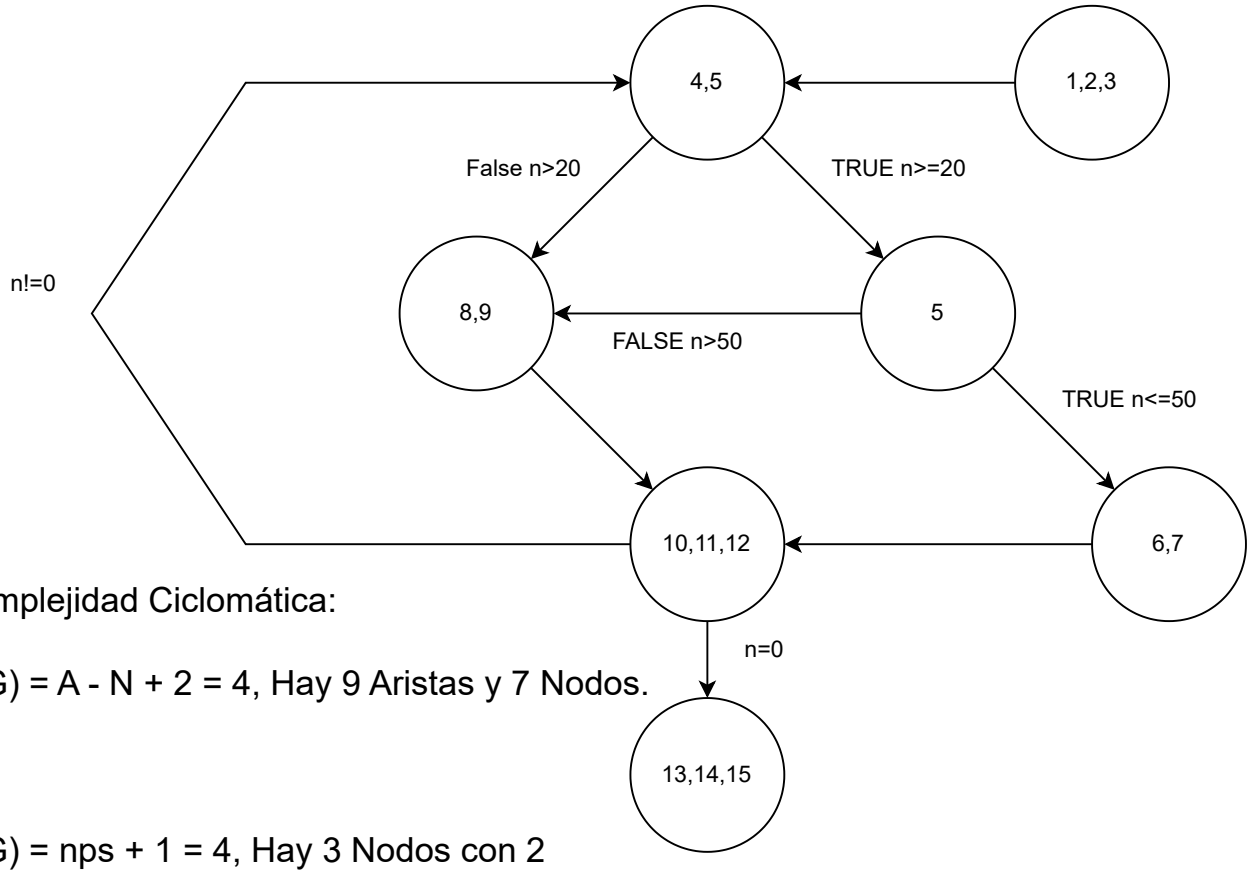
Programa Simple sin Mucho Riesgo.

Ejercicio - 4

```
1  function obtener_media : real ;

   var
       n, suma, conta, suma2, total_num : integer ;

2  begin
3      read( n ) ;
4      repeat
5          if (n >= 20 and n <= 50) then
6              suma := suma + n ;
7              conta := conta + 1
8          else
9              suma2 := suma2 + n ;
10         total_num := total_num + 1 ;
11         read (n) ;
12     until n = 0 ;
13     obtener_media := suma / conta ;
14     write (total_num, suma2) ;
15 end ;
```



Complejidad Ciclomática:

V(G) = A - N + 2 = 4, Hay 9 Aristas y 7 Nodos.

ó

V(G) = nps + 1 = 4, Hay 3 Nodos con 2 Aristas.

Programa Simple sin Mucho Riesgo.

Ejercicio - 5

```
public static double ReturnAverage(int value[],
                        int AS, int MIN, int MAX){
  /*
   Function: ReturnAverage  Computes the  average
   of all  those  numbers in the  input array  in
   the  positive  range  [MIN, MAX]. The  maximum
   size  of the array is AS. But, the  array size
   could be smaller than AS in which case the end
   of input is represented by -999.
  */
1    int i, ti, tv, sum;
2    double av;
3    i = 0; ti = 0; tv = 0; sum = 0;
4    while (ti < AS && value[i] != -999) {
5        ti++;
6        if (value[i] >= MIN && value[i] <= MAX) {
7           tv++;
8           sum = sum + value[i];
         }
9        i++;
     }
10   if (tv > 0)
11      av = (double)sum/tv;
12   else
13      av = (double) -999;
14   return (av);
}
```

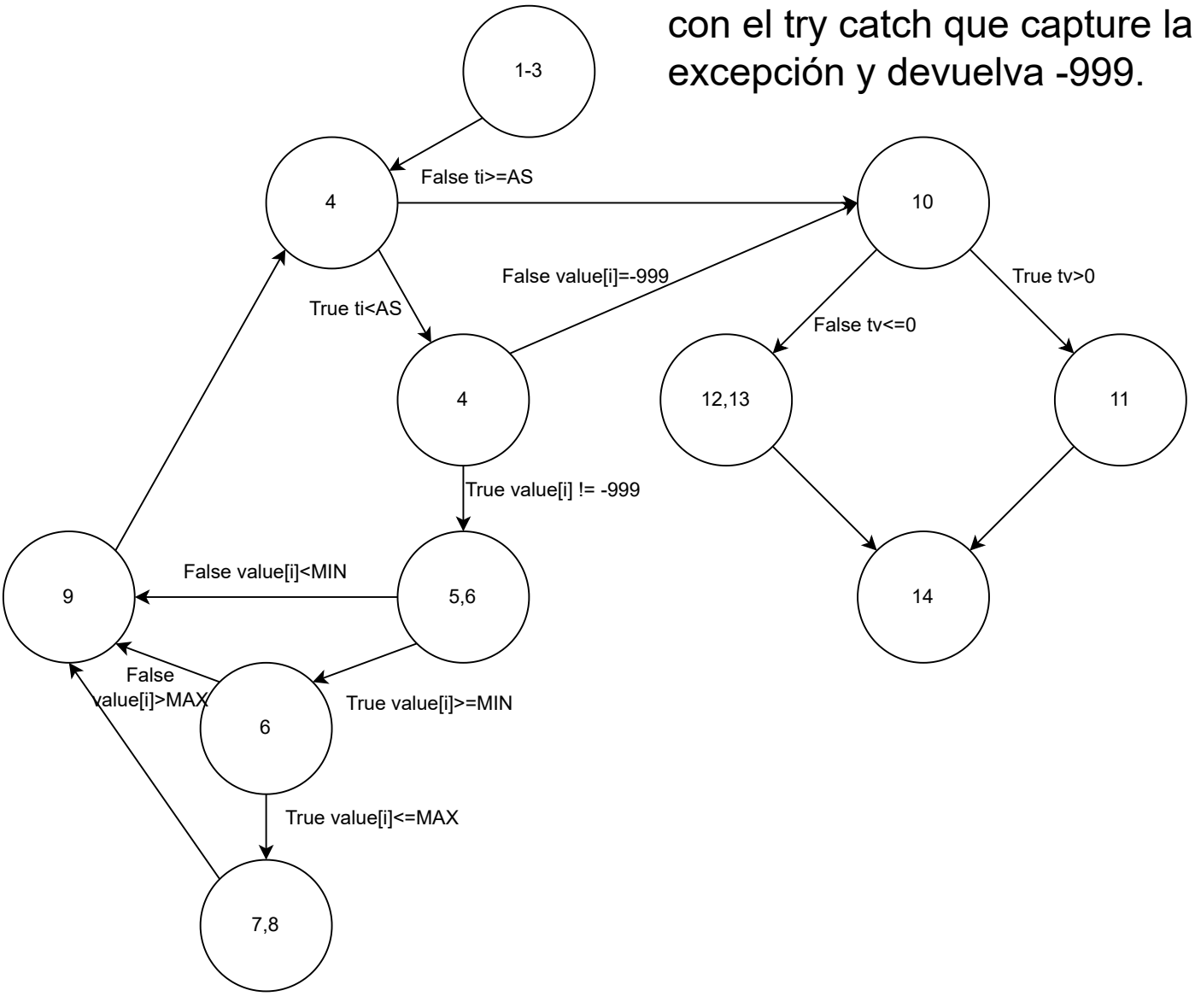
Complejidad Ciclomática:

V(G) = A - N + 2 = 6 Hay 15 Aristas y 11 Nodos.

V(G) = nps + 1 = 6 Hay 5 Nodos con 2 Aristas.

Error de Lógica:

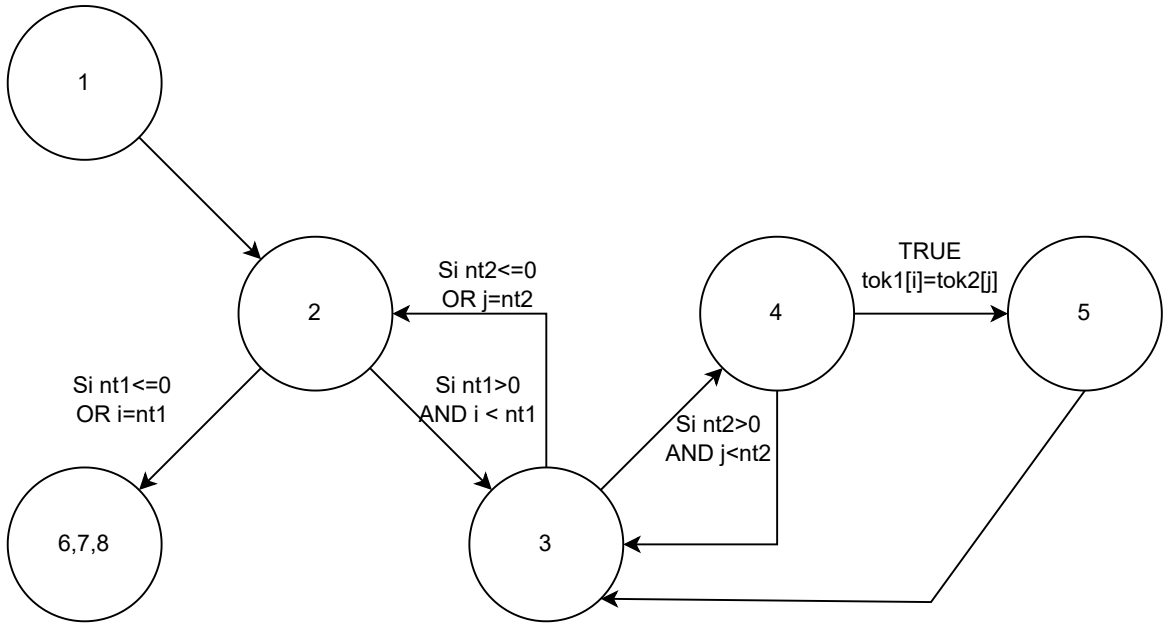
El Código tiene un Error de Lógica, ya que si se le pasa un tamaño de Array mayor al tamaño real, cuando busque en una posición que no existe, dará un excepción de Index Out of Bounds del Array. Falta Implementar el Codigo con el try catch que capture la excepción y devuelva -999.



Ejercicio - 6

```
Calcula cohesión (int nt1, nt2; String tok1[ ], tok2[ ])

1 numAdh = 0
2 Para i de 0 hasta nt1-1
3     Para j de 0 hasta nt2-1
4         Si tok1[i]=tok2[j] entonces
5             numAdh = numAdh +1
6 total = tok1 + tok2 – numAdh
7 cohesión = numAdh / total
8 regresa cohesión
```



Complejidad Ciclomática:

V(G) = A - N + 2 = 4 Hay 8 Aristas y 6 Nodos.

ó

V(G) = nps + 1 = 4 Hay 3 Nodos con 2 Aristas.

Programa Simple sin Mucho Riesgo