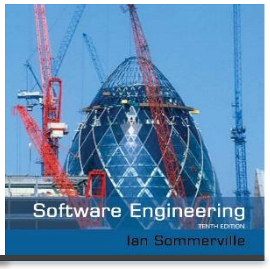
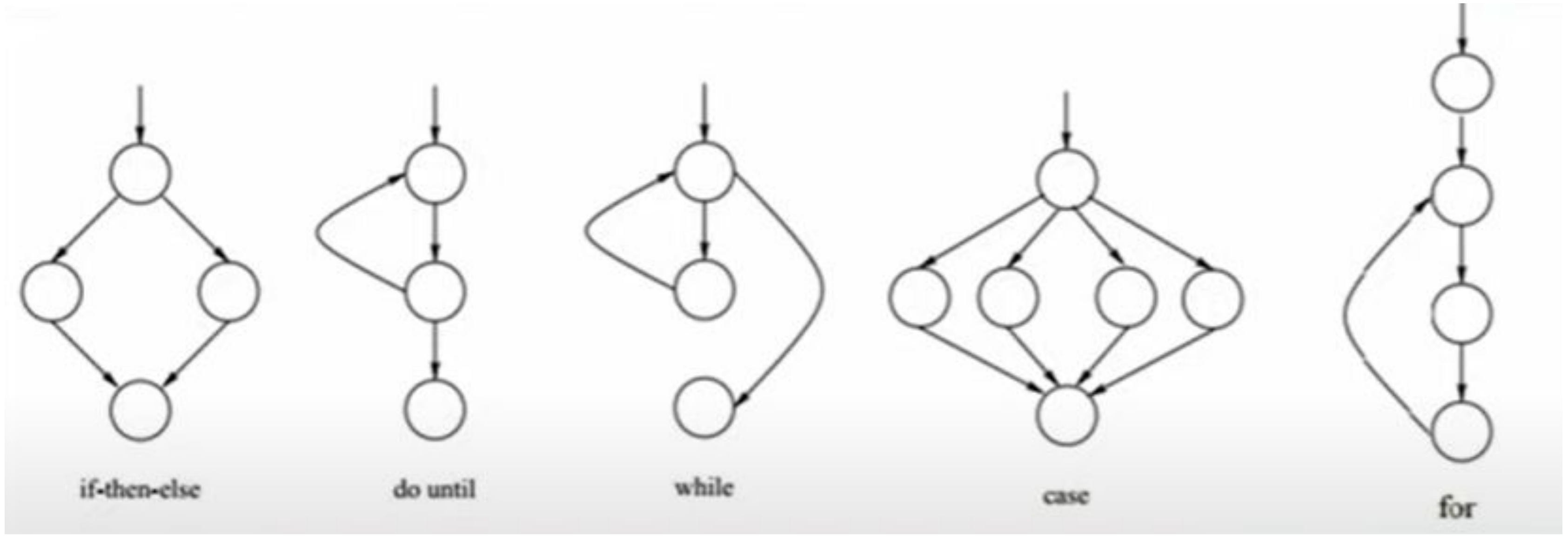

Software Engineering
TENTH EDITION
Ian Sommerville

Basis Path Testing

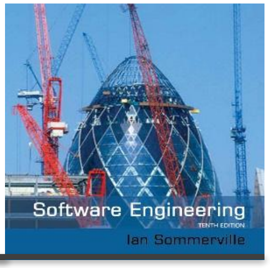
Control flow graph symbols & CC formula



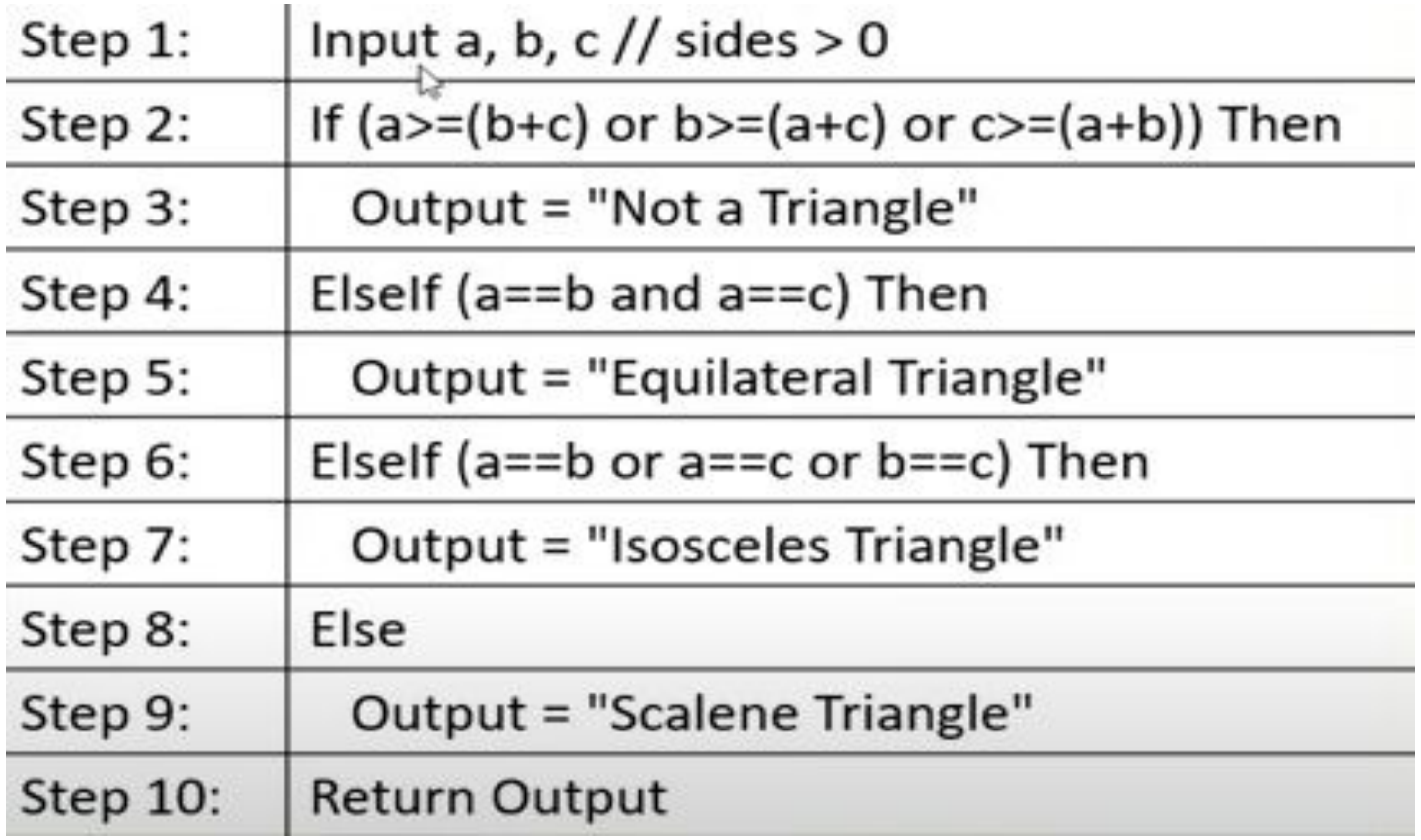
CC = E – N + 2P

Basis Path testing

1. Draw the Control Flow Graph
2. Calculate the Cyclomatic complexity using all the methods
3. List all the Independent Paths
4. Design test cases from independent paths

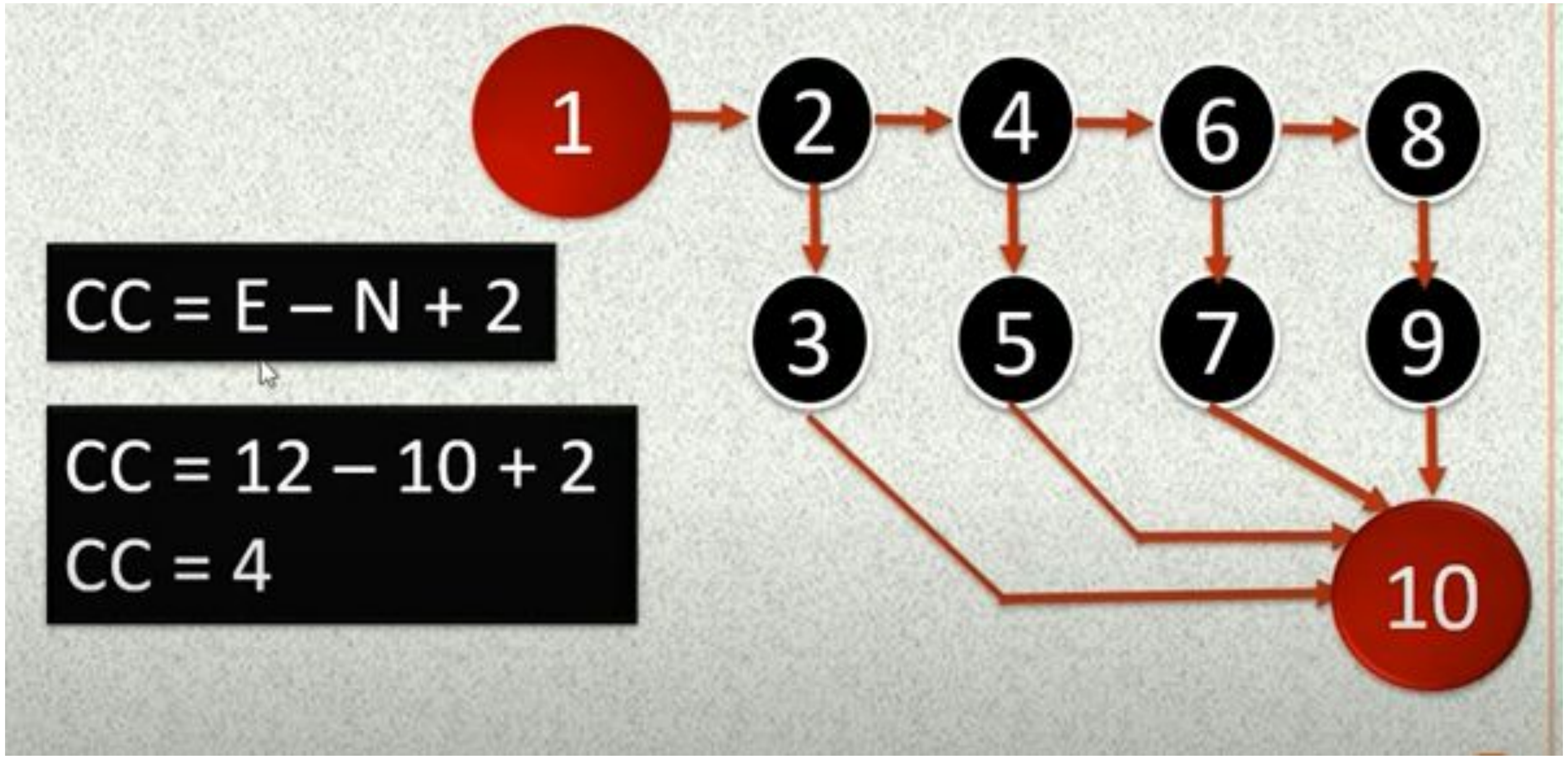


Example 1 -- Step-1: label steps



Step 1:	Input a, b, c // sides > 0
Step 2:	If (a>=(b+c) or b>=(a+c) or c>=(a+b)) Then
Step 3:	Output = "Not a Triangle"
Step 4:	ElseIf (a==b and a==c) Then
Step 5:	Output = "Equilateral Triangle"
Step 6:	ElseIf (a==b or a==c or b==c) Then
Step 7:	Output = "Isosceles Triangle"
Step 8:	Else
Step 9:	Output = "Scalene Triangle"
Step 10:	Return Output

Step-2: create graph

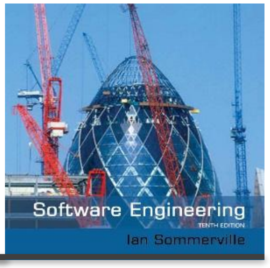


CC = E – N + 2

CC = 12 – 10 + 2

CC = 4

Step-3: select paths



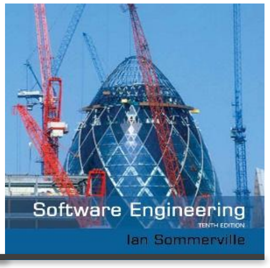
1-2-3-10

1-2-4-5-10

1-2-4-6-7-10

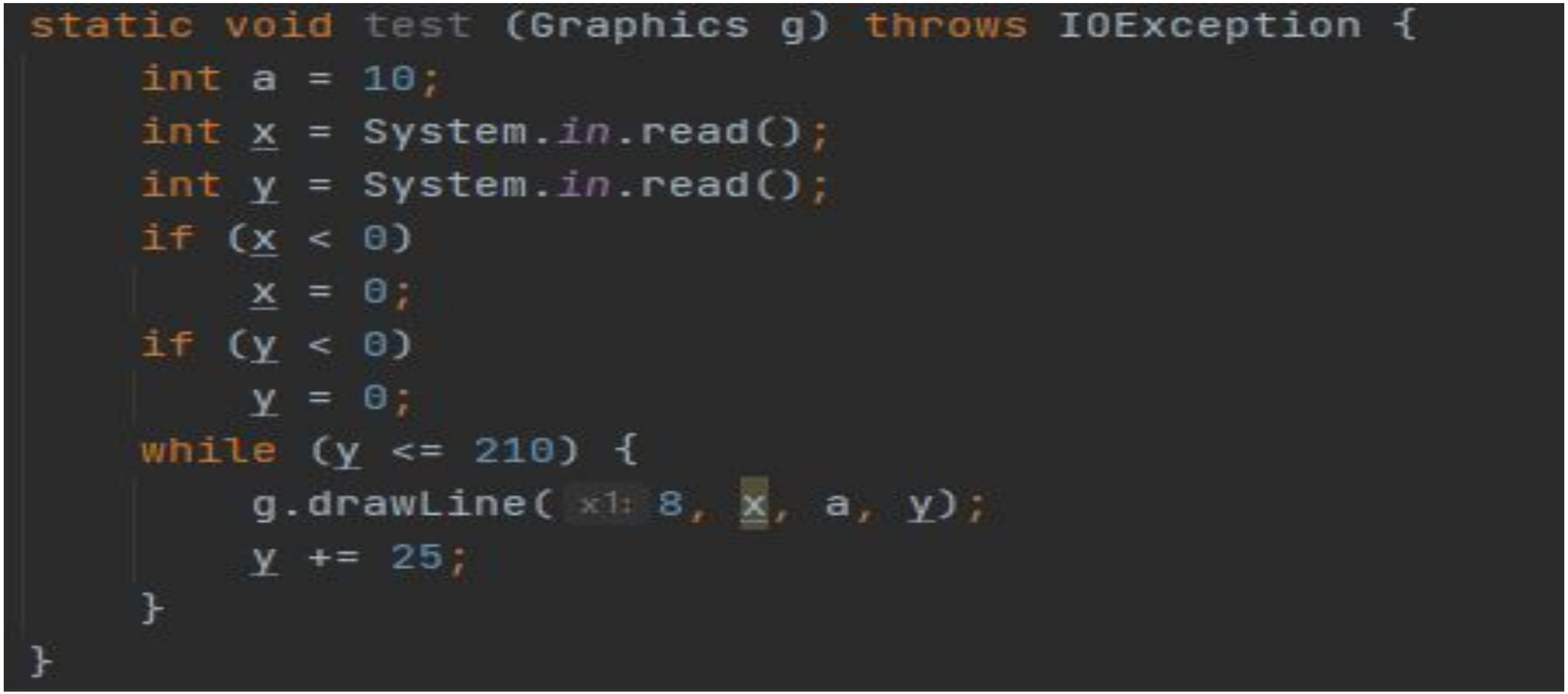
1-2-4-6-8-9-10

Step-3: create tests



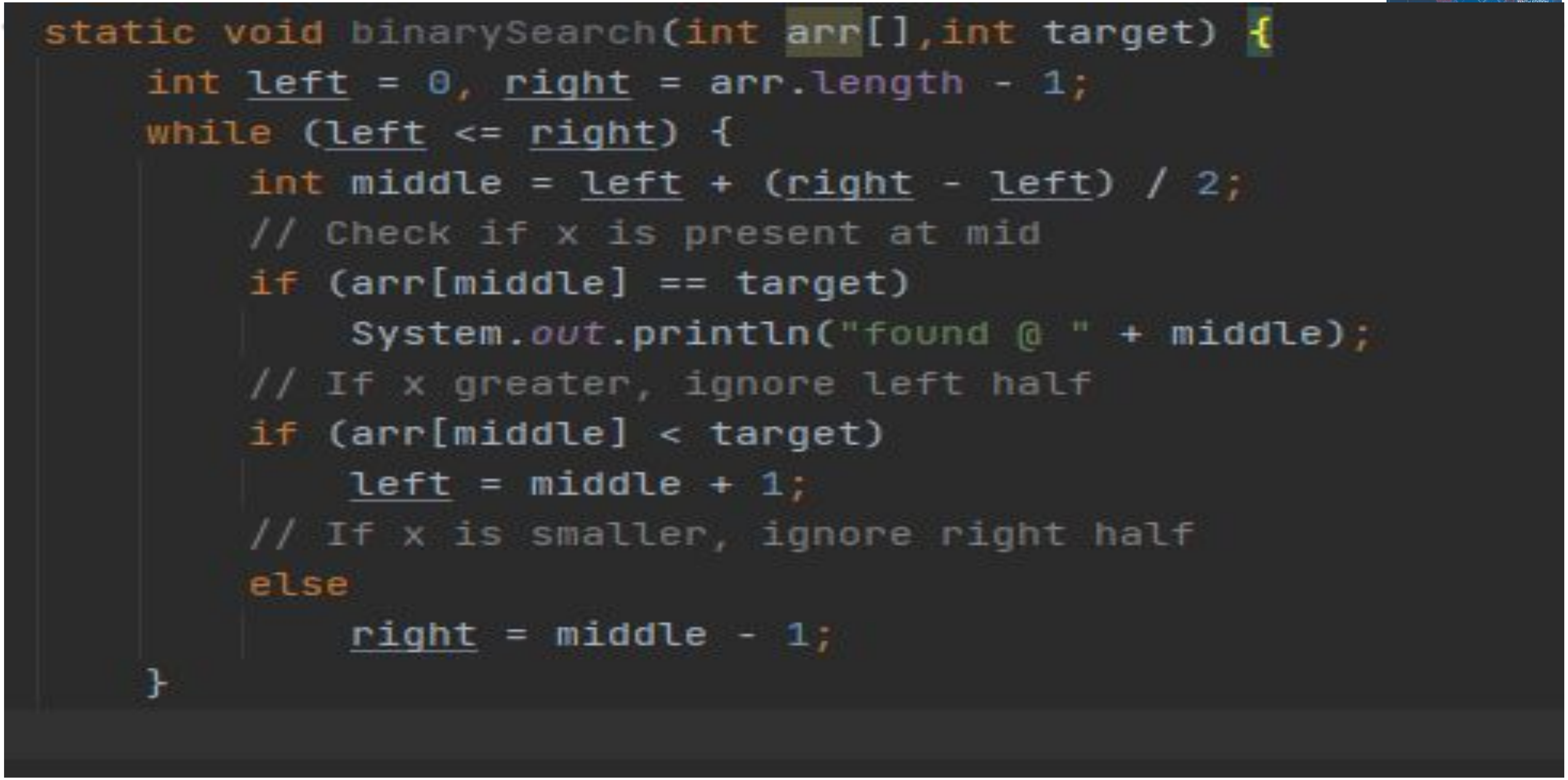
If	Else if	Else if	Else	Expected outcome
T	F	F	F	Not a triangle
F	T	F	F	Equilateral
F	F	T	F	Isosceles
F	F	F	T	Scalene

Example 2: Create control flow graph & calculate Cyclomatic Complexity



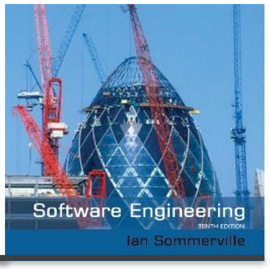
```
static void test (Graphics g) throws IOException {
    int a = 10;
    int x = System.in.read();
    int y = System.in.read();
    if (x < 0)
        x = 0;
    if (y < 0)
        y = 0;
    while (y <= 210) {
        g.drawLine( x1: 8, x, a, y);
        y += 25;
    }
}
```

Example 3: Create control flow graph & identify paths



```
static void binarySearch(int arr[],int target) {
    int left = 0, right = arr.length - 1;
    while (left <= right) {
        int middle = left + (right - left) / 2;
        // Check if x is present at mid
        if (arr[middle] == target)
            System.out.println("found @ " + middle);
        // If x greater, ignore left half
        if (arr[middle] < target)
            left = middle + 1;
        // If x is smaller, ignore right half
        else
            right = middle - 1;
    }
```

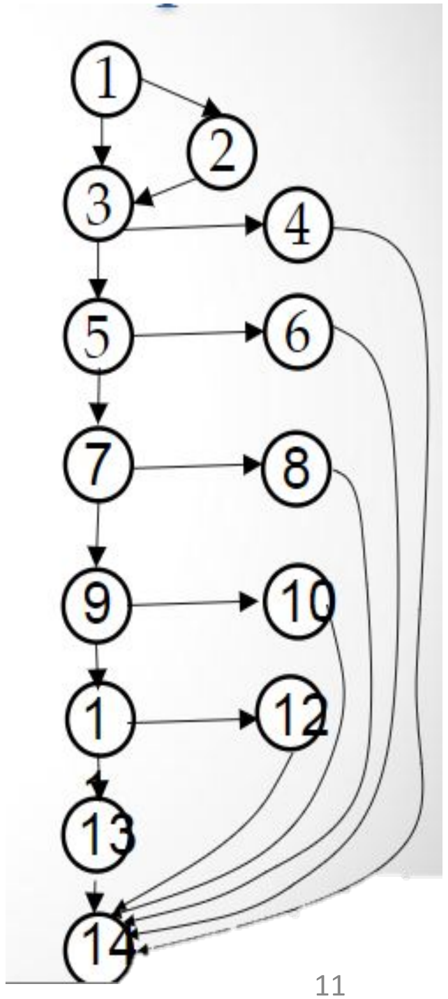
Example 4: perform basis path testing for following code



```
public double calculate(int amount)
   {
-1-  double rushCharge = 0;
-1-  if (nextday.equals("yes") )
    {
-2-       rushCharge = 14.50;
    }
-3-  double tax = amount * .0725;
-3-  if (amount >= 1000)
    {
-4-       shipcharge = amount * .06 + rushCharge;
    }
-5-  else if (amount >= 200)
    {
-6-       shipcharge = amount * .08 + rushCharge;
    }
-7-  else if (amount >= 100)
    {
-8-       shipcharge = 13.25 + rushCharge;
    }
-9-  else if (amount >= 50)
    {
-10-      shipcharge = 9.95 + rushCharge;
    }
-11- else if (amount >= 25)
    {
-12-      shipcharge = 7.25 + rushCharge;
    }
    else
    {
-13-      shipcharge = 5.25 + rushCharge;
    }
-14- total = amount + tax + shipcharge;
-14- return total;
    } //end calculate
```

Step 1 & 2: label code and create control flow graph

```
public double calculate(int amount)
   {
-1-  double rushCharge = 0;
-1-  if (nextday.equals("yes") )
     {
-2-       rushCharge = 14.50;
     }
-3-  double tax = amount * .0725;
-3-  if (amount >= 1000)
     {
-4-       shipcharge = amount * .06 + rushCharge;
     }
-5-  else if (amount >= 200)
     {
-6-       shipcharge = amount * .08 + rushCharge;
     }
-7-  else if (amount >= 100)
     {
-8-       shipcharge = 13.25 + rushCharge;
     }
-9-  else if (amount >= 50)
     {
-10-      shipcharge = 9.95 + rushCharge;
     }
-11- else if (amount >= 25)
     {
-12-      shipcharge = 7.25 + rushCharge;
     }
     else
     {
-13-      shipcharge = 5.25 + rushCharge;
     }
-14- total = amount + tax + shipcharge;
-14- return total;
     } //end calculate
```



Step 3: identify paths

P-01: 1, 2, 3, 5,7,9,11,13,14

P-02: 1,3,4,14

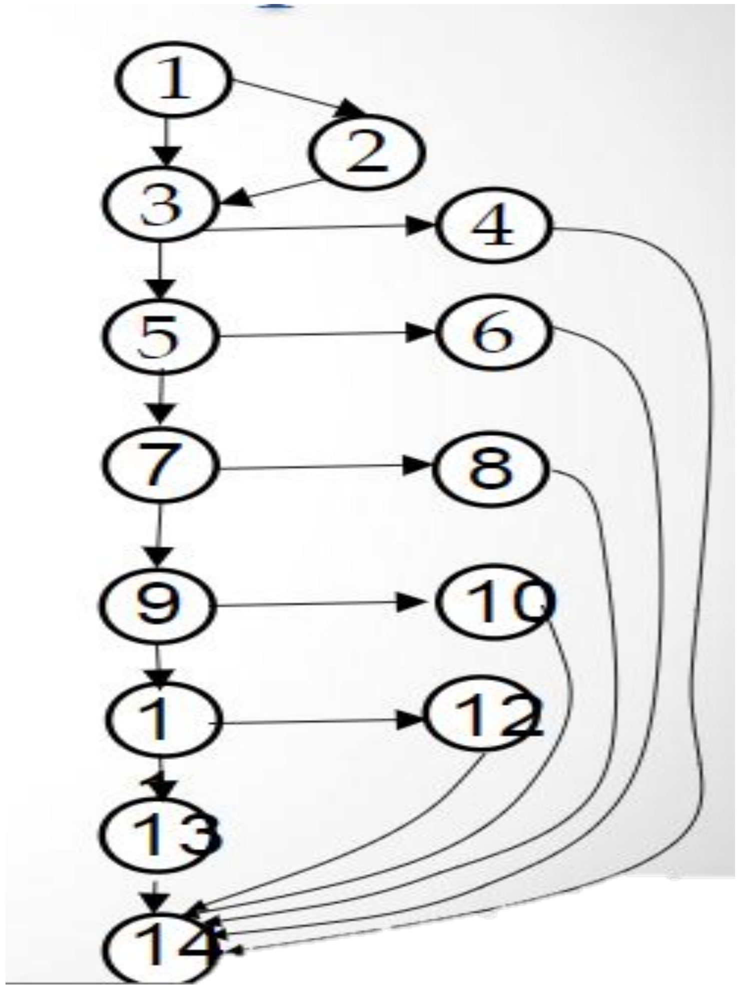
P-03: 1,3,5,6,14

P-04: 1,3,5,7,8,14

P-05: 1,3,5,7,9,10,14

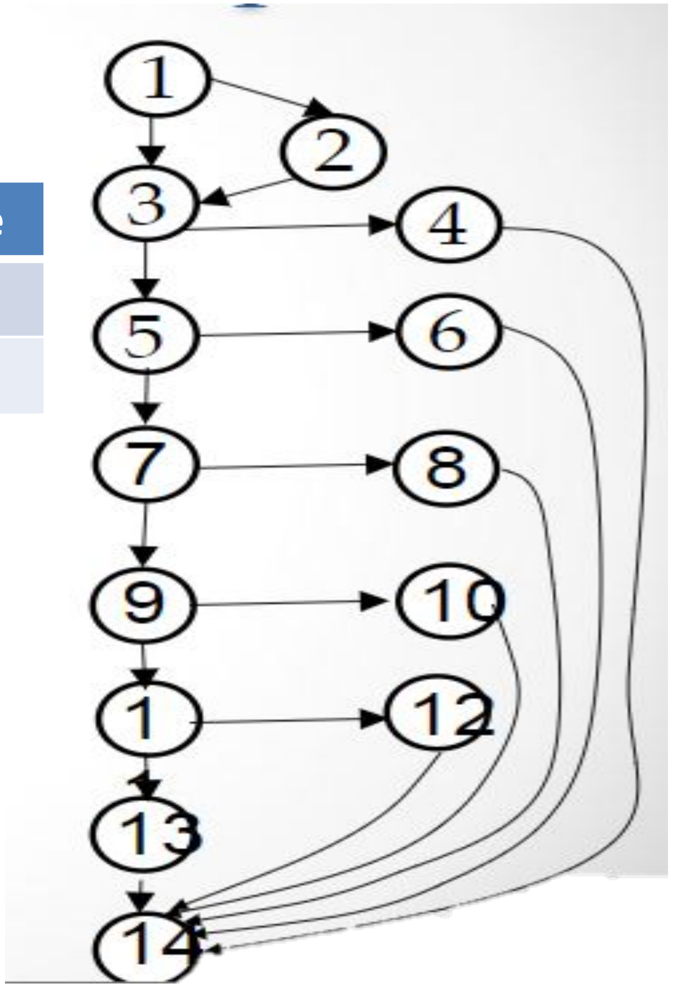
P-06: 1,3,5,7,9,11,12,14

P-07: 1,3,5,7,9,11,13,14



Step 4: define test case

Path #	Amount	NextDay	Expected outcome
2	1500	Yes	?
1	10	Yes	?


1
2
3
4
5
6
7
8
9
10
1
12
13
14