Is It Xmas?

Source Code

```
1  def is_it_xmas?(month, day)
2      if month == 12 && day == 25
3          return true
4      else
5          return false
6      end
7  end
```

Example Test Suite

Test #	Input		Expected Output
	month	day	
1	12	25	true
2	1	1	false

Buggy Variant of Source Code

```
2      if month == 12 || day == 25
```

For more practice, try answering the questions with this alternative example test suite.

Alternative Example Test Suite

Test #	Input		Expected Output
	month	day	
1	12	25	true
2	12	26	false

Min of Three

Source Code

```
 1  def min_of_three(x, y, z)
 2      if x < y then
 3          if x < z then
 4              return x
 5          else
 6              return z
 7          end
 8      else
 9          if y < z then
10              return y
11          else
12              return z
13          end
14      end
15  end
```

Example Test Suite

Test #	Input			Expected Output
	x	y	z	
1	1	2	3	1
2	1	2	0	0
3	2	1	3	1
4	2	1	0	0

Buggy Variant of Source Code

```
 8      else
 9          if y < z then
10              return x
```

For more practice, try answering the questions with this alternative example test suite.

Alternative Example Test Suite

Test #	Input			Expected Output
	x	y	z	
1	0	0	0	0
2	1	0	0	0
3	0	1	0	0
4	0	0	1	0

Greatest Common Divisor

Source Code

```
 1  def gcd(x, y)
 2      if x == 0
 3          return y
 4      end
 5      if y == 0
 6          return x
 7      end
 8      while x != y
 9          if x > y
10              x = x - y
11          else
12              y = y - x
13          end
14      end
15      return x
16  end
```

Example Test Suite

Test #	Input		Expected Output
	x	y	
1	0	1	1
2	1	0	1
3	1	1	1
4	2	1	1
5	1	2	1

Buggy Variant of Source Code

```
 8      while x != 1
```

For more practice, try answering the questions with this alternative example test suite.

Alternative Example Test Suite

Test #	Input		Expected Output
	x	y	
1	0	1	1
2	1	0	1
3	2	1	1
4	1	2	1

Binary Search

Source Code

```
 1  def binary_search(array, key)
 2      imin = 0
 3      imax = array.length - 1
 4      while imin <= imax
 5          imid = (imin + ((imax - imin) / 2)).to_i
 6          if array[imid] == key
 7              return imid
 8          elsif array[imid] < key
 9              imin = imid + 1
10          else
11              imax = imid - 1
12          end
13      end
14      return -1
15  end
```

Example Test Suite

Test #	Input		Expected Output
	array	key	
1	[ ]	'a'	-1
2	['a']	'b'	-1
3	['b']	'a'	-1
4	['a', 'b', 'c']	'b'	1

Buggy Variant of Source Code

```
 2      imin = 1
```

For more practice, try answering the questions with this alternative example test suite.

Alternative Example Test Suite

Test #	Input		Expected Output
	array	key	
1	['a', 'b', 'c', 'd', 'e', 'f', 'g']	'c'	2

Problems

Answer the following questions for each of the above example methods and test suites.

Control-Flow Graph: Draw a control-flow graph for the function. In addition to the usual CFG features, label each node with the corresponding code-line number.

Statement Coverage: For each test in the test suite, list the nodes covered by the test case.

Test 1) _____

Test 2) _____

Test 3) _____

Test 4) _____

Test 5) _____

Test 6) _____

Does the test suite achieve **statement coverage**? Yes No

If "No", which nodes did the test suite miss? _____

Branch Coverage: For each test in the test suite, list the relevant edges covered by the test case. Denote an edge like this, 2→3, which denotes the edge from node 2 to node 3.

Test 1) _____

Test 2) _____

Test 3) _____

Test 4) _____

Test 5) _____

Test 6) _____

Does the test suite achieve **branch coverage**? Yes No

If "No", which edges did the test suite miss? _____

Path Coverage: First, list all the paths through the CFG. For each path, list the sequence of nodes on the path, like this: 2→3→5→6, which denotes a path from node 2 to node 6. You need only cover executions that involve at most 1 iteration of each loop (if there are any loops). There may be more lines below than there are paths; use only as many lines as you need.

For each test in the test suite, list the path covered by the test case.

Test 1) _____

Test 2) _____

Test 3) _____

Test 4) _____

Test 5) _____

Test 6) _____

Does the test suite achieve **path coverage**? Yes No

If "No", which paths did the test suite miss? _____

Would the example test suite detect the bug in the buggy variant of the code? If so, which test(s) would fail?