**13. Perform white box testing – Cyclomatic complexity, data flow testing, control flow testing for the given problem statement: Write algorithm for deciding whether the entered number is prime number or not**

**Algorithm to Check if a Number is Prime**

java

public boolean isPrime(int number) {

if (number <= 1) {

return false;

}

for (int i = 2; i <= Math.sqrt(number); i++) {

if (number % i == 0) {

return false;

}

}

return true;

}

**1. Cyclomatic Complexity**

Cyclomatic complexity is a metric used to measure the complexity of a program by counting the number of linearly independent paths through the program's source code.

**Calculation:**

* **Number of Decisions (D)**: 2 (if conditions)
* **Cyclomatic Complexity (CC)**: D + 1 = 2 + 1 = 3

The cyclomatic complexity of the isPrime function is 3, indicating there are 3 independent paths through the function.

**2. Data Flow Testing**

Data Flow Testing focuses on the points where variables receive values and where these values are used.

**Variables:**

* number: Input parameter to the function
* i: Loop counter

**Key Points:**

* Definition: number, i
* Usage:
  + number is used in two if conditions.
  + i is defined at the start of the loop and used in the loop condition and within the loop body.

**3. Control Flow Testing**

Control Flow Testing involves analyzing the flow of the program to ensure all possible paths are tested.

**Control Flow Graph:**

1. **Start**: Enter function
2. **Node 1**: Check if (number <= 1)
   * **True**: Go to Node 2
   * **False**: Go to Node 3
3. **Node 2**: Return false
4. **Node 3**: Enter for loop, i = 2
5. **Node 4**: Check i <= Math.sqrt(number)
   * **True**: Go to Node 5
   * **False**: Go to Node 7
6. **Node 5**: Check if (number % i == 0)
   * **True**: Go to Node 6
   * **False**: Go to Node 4 (increment i)
7. **Node 6**: Return false
8. **Node 7**: Return true

**Test Cases**

**Path 1: Not Prime (number <= 1)**

* Input: number = 0
* Expected Output: false

**Path 2: Prime Number (number = 2)**

* Input: number = 2
* Expected Output: true

**Path 3: Not Prime (number = 4)**

* Input: number = 4
* Expected Output: false

**Path 4: Prime Number (number = 17)**

* Input: number = 17
* Expected Output: true

**Path 5: Not Prime (number = 20)**

* Input: number = 20
* Expected Output: false