“One of the eternally discussed problems is ... communications ... in programming activities ... consider what can be done ... in those areas where automation is not taking care of the basic documentation problem.”

Melba Mouton
Head Computer Programmer
Program Production Section Chief
Goddard Space Flight Center, NASA
What is output by the following method?

```java
public static void mysteryB(boolean b) {
    System.out.print(b + " ");
    b = (b == false);
    System.out.print(b);
}
```

A. no output due to syntax error
B. no output due to runtime error
C. not possible to predict output
D. always outputs true false OR false true
E. always outputs true true OR false false
Assertions

- **Assertion**: A declarative sentence that is either true or false

- Examples:
  2 + 2 equals 4
  The St. Louis Cardinals played in the 2011 world series
  x > 45
  It is raining.
  UT beat OU last year.
  UT qualified for the NCAA tourney last year.

- Not assertions
  How old are you?
  Take me to H.E.B.
Some assertions are true or false depending on context. Which of these depend on the context?

- 2 + 2 equals 4
- The St. Louis Cardinals played in the world series this year
- x > 45
- It is raining.
- UT beat OU last year.
- UT qualified for the NCAA tourney last year.
Assertions

Another important skill in programming and computer science is the ability "to make assertions about your programs and to understand the contexts in which those assertions will be true."

Scanner console = new Scanner(System.in);
System.out.print("Enter Y or N: ");
String result = console.nextLine();
// is result equal to "Y" or "N" here?
Checking Input

Scanner console = new Scanner(System.in);
System.out.print("Enter Y or N: ");
String result = console.nextLine();
while(!result.equals("Y") && !result.equals("N")){
    System.out.print("That wasn't a Y or N. ");
    System.out.print("Enter Y or N: ");
    result = console.nextLine();
}
// is result equal to "Y" or "N" here?
Assertions

- **Provable Assertion**: An assertion that can be proven to be true at a particular point in program execution.

- **Program Verification**: A field of computer science that involves reasoning about the formal properties of programs to prove the correctness of a program.
  - Instead of testing.
  - A number of UTCS faculty are involved in verification research: Emerson, Hunt, Lam, Moore, Young
Reasoning about assertions

Suppose you have the following code:

```java
if (x > 3) {
    // Point A
    x--;  
} else {
    // Point B
    x++;  
    // Point C
}
// Point D
```

What do you know about `x`'s value at the three points?
- Is `x > 3`? Always? Sometimes? Never?
Assertions in code

- We can make assertions about our code and ask whether they are true at various points in the code.
  - Valid answers are ALWAYS, NEVER, or SOMETIMES.

```java
System.out.print("Type a nonnegative number: ");
double number = console.nextDouble();
// Point A: is number < 0.0 here? (SOMETIMES)

while (number < 0.0) {
    // Point B: is number < 0.0 here? (ALWAYS)
    System.out.print("Negative; try again: ");
    number = console.nextDouble();
    // Point C: is number < 0.0 here? (SOMETIMES)
}

// Point D: is number < 0.0 here? (NEVER)
```
Reasoning about programs

- Right after a variable is initialized, its value is known:
  ```java
  int x = 3;
  // is x > 0?  ALWAYS
  ```

- In general you know nothing about parameters' values:
  ```java
  public static void mystery(int a, int b) {
  // is a == 10?  SOMETIMES
  ```
Reasoning about programs

- But inside an `if`, `while`, etc., you may know something:

```java
public static void mystery(int a, int b) {
    if (a < 0) {
        // is a == 10? NEVER
        ...
    }
}
```
Assertions and loops

- At the start of a loop's body, the loop's test must be **true**:
  ```java
  while (y < 10) {
      // is y < 10? ALWAYS
      ...
  }
  ```

- After a loop, the loop's test must be **false**:
  ```java
  while (y < 10) {
      ...
  }
  // is y < 10? NEVER
  ```

- Inside a loop's body, the loop's test may become **false**:
  ```java
  while (y < 10) {
      y++;
      // is y < 10? SOMETIMES
  }
  ```
"Sometimes"

- Things that cause a variable's value to be unknown (often leads to "sometimes" answers):
  - reading from a Scanner
  - reading a number from a Random object
  - initial value of a parameter in a method
public static void mystery(int x, int y) {
    int z = 0;

    // Point A
    while (x >= y) {
        // Point B
        x = x - y;
        z++;
        if (x != y) {
            // Point C
            z = z * 2;
        }
    }

    // Point D
}

// Point E
System.out.println(z);

For each assertion state if it is ALWAYS, NEVER, or SOMETIMES true at the specified points in the code.

<table>
<thead>
<tr>
<th></th>
<th>x &lt; y</th>
<th>x == y</th>
<th>z == 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point A</td>
<td>ALWAYS</td>
<td>SOMETIMES</td>
<td>ALWAYS</td>
</tr>
<tr>
<td>Point B</td>
<td>NEVER</td>
<td>SOMETIMES</td>
<td>SOMETIMES</td>
</tr>
<tr>
<td>Point C</td>
<td>SOMETIMES</td>
<td>NEVER</td>
<td>NEVER</td>
</tr>
<tr>
<td>Point D</td>
<td>SOMETIMES</td>
<td>SOMETIMES</td>
<td>NEVER</td>
</tr>
<tr>
<td>Point E</td>
<td>ALWAYS</td>
<td>NEVER</td>
<td>SOMETIMES</td>
</tr>
</tbody>
</table>
public static int mystery(Scanner console) {
    int prev = 0;
    int count = 0;
    int next = console.nextInt();

    // Point A
    while (next != 0) {
        // Point B
        if (next == prev) {
            // Point C
            count++;  
        }
        prev = next;
        next = console.nextInt();
        // Point D
    }
    // Point E
    return count;
}

For each assertion state if it is
ALWAYS, NEVER, or SOMETIMES true
at the specified points in the code.

<table>
<thead>
<tr>
<th>Point</th>
<th>next == 0</th>
<th>prev == 0</th>
<th>next == prev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point A</td>
<td>SOMEBEtimes</td>
<td>ALWAYS</td>
<td>SOMEBEtimes</td>
</tr>
<tr>
<td>Point B</td>
<td>NEVER</td>
<td>SOMEBEtimes</td>
<td>SOMEBEtimes</td>
</tr>
<tr>
<td>Point C</td>
<td>NEVER</td>
<td>NEVER</td>
<td>ALWAYS</td>
</tr>
<tr>
<td>Point D</td>
<td>SOMEBEtimes</td>
<td>NEVER</td>
<td>SOMEBEtimes</td>
</tr>
<tr>
<td>Point E</td>
<td>ALWAYS</td>
<td>SOMEBEtimes</td>
<td>SOMEBEtimes</td>
</tr>
</tbody>
</table>
// Assumes y >= 0, and returns x^y
public static int pow(int x, int y) {
    int prod = 1;

    // Point A
    while (y > 0) {
        // Point B
        if (y % 2 == 0) {
            // Point C
            x = x * x;
            y = y / 2;
            // Point D
        } else {
            // Point E
            prod = prod * x;
            y--;
            // Point F
        }
    }
    // Point G
    return prod;
}

For each assertion state if it is ALWAYS, NEVER, or SOMETIMES true at the specified points in the code.

<table>
<thead>
<tr>
<th>Point</th>
<th>y &gt; 0</th>
<th>y % 2 == 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point A</td>
<td>SOMETIMES</td>
<td>SOMETIMES</td>
</tr>
<tr>
<td>Point B</td>
<td>ALWAYS</td>
<td>SOMETIMES</td>
</tr>
<tr>
<td>Point C</td>
<td>ALWAYS</td>
<td>ALWAYS</td>
</tr>
<tr>
<td>Point D</td>
<td>ALWAYS</td>
<td>SOMEBTIMES</td>
</tr>
<tr>
<td>Point E</td>
<td>ALWAYS</td>
<td>NEVER</td>
</tr>
<tr>
<td>Point F</td>
<td>SOMETIMES</td>
<td>ALWAYS</td>
</tr>
<tr>
<td>Point G</td>
<td>NEVER</td>
<td>ALWAYS (SOMETIMES if expectation ignored)</td>
</tr>
</tbody>
</table>