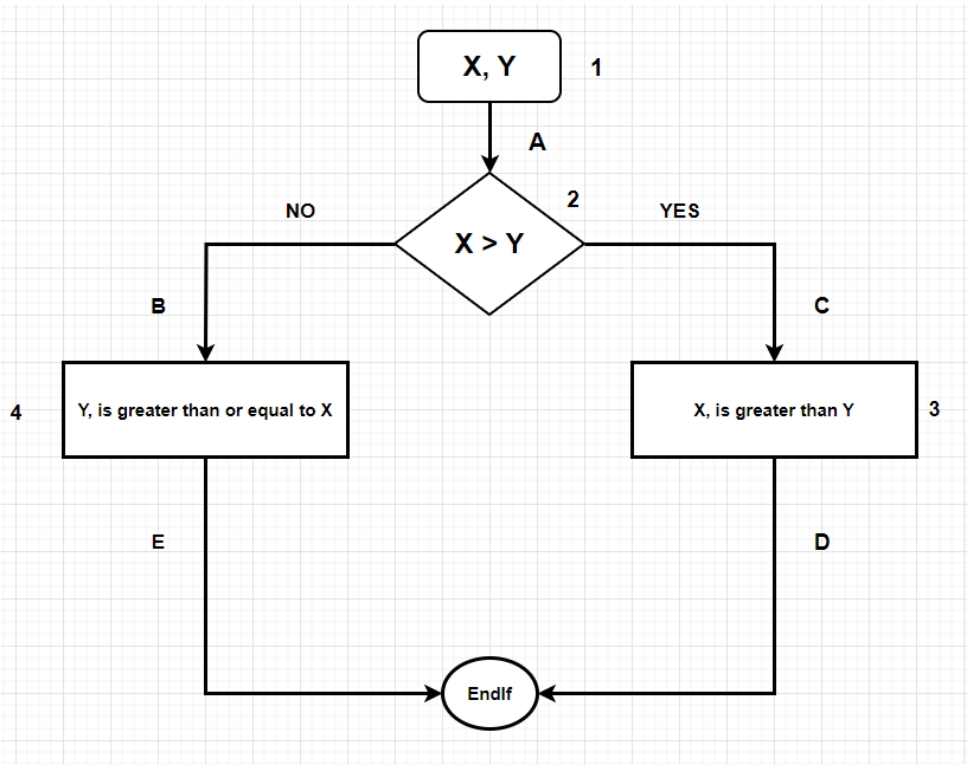
Task 1:
Display the pseudocode below in the form of a flowchart.

```
Begin
Input X, Y
If X > Y
__Print (X, 'is greater than', Y)
Else
__Print (Y, is greater than or equal to', X)
EndIf
End
```

What is the minimum number of test cases required to guarantee 100% statement and 100% decision coverage?

A. Statement coverage = 3, Decision coverage = 3
B. Statement coverage = 2, Decision coverage = 2
C. Statement coverage = 1, Decision coverage = 2
D. Statement coverage = 2, Decision coverage = 1



Statement coverage means we must go through all the blocks from the input to the output — that is, we need to cover all the statements in the code.

Decision coverage means we must verify all the conditions at least once — both when they are true and when they are false.

The correct answer for Task 1 is:

B. Statement coverage = 2, Decision coverage = 2

Statement coverage:

- We follow the path: 1 → 2 → 3 → EndIf
- We follow the path: 1 → 2 → 4 → EndIf

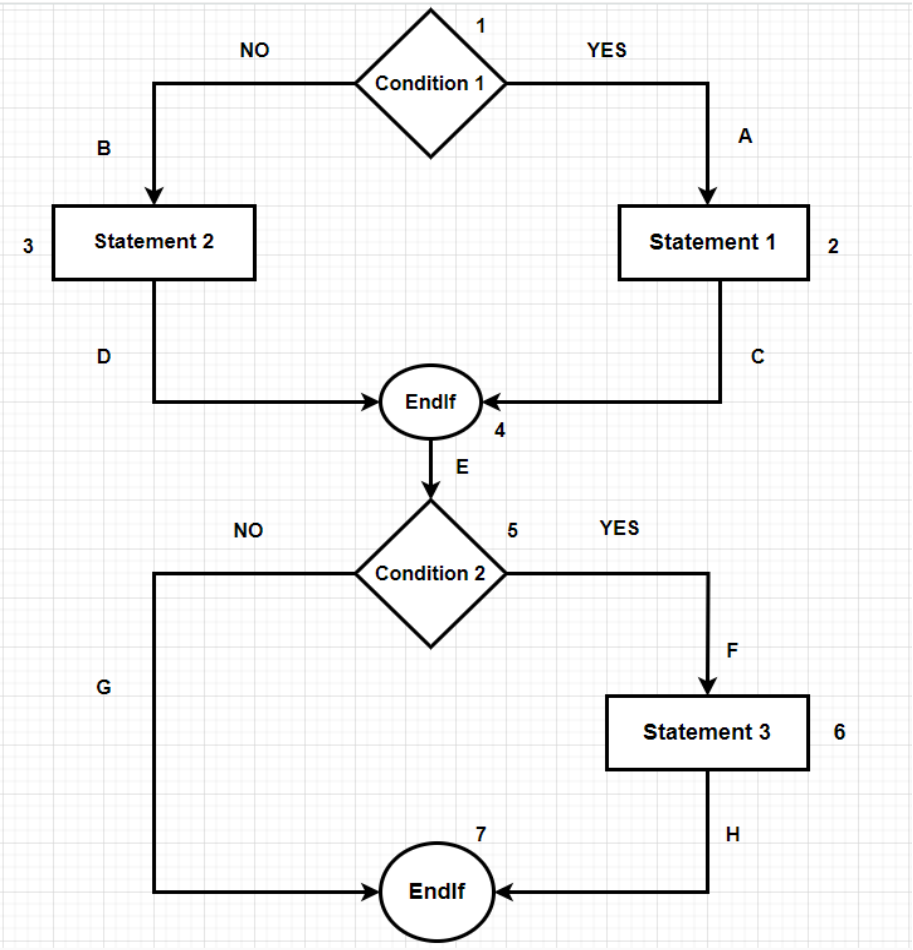
Decision coverage:

- We follow the path: 1 → A → 2 → Yes → C → 3 → D → EndIf
- We follow the path: 1 → A → No → B → 4 → E → EndIf

Task 2: Display the pseudocode below as a flowchart.

```
if (Condition 1)
then statement 1
else statement 2
fi
if (Condition 2)
then statement 3
fi
```

What is the minimum number of test cases required to guarantee 100% path coverage?
A. 1
B. 2
C. 3
D. None of the answers is correct



Path coverage: all paths/conditions must be checked, both True with False and False with True.

Answer for Task 2: D. None of the answers is correct

To achieve **100% path coverage**, all unique execution paths through the flow must be tested. The following are the required paths:

1. 1 → YES-A → 2 → C → 4-EndIf → E → 5 → YES-F → 6 → H → 7-EndIf
2. 1 → NO-B → 3 → D → 4-EndIf → E → 5 → NO-G → 7-EndIf
3. 1 → NO-B → 3 → D → 4-EndIf → E → 5 → YES-F → 6 → H → 7-EndIf
4. 1 → YES-A → 2 → C → 4-EndIf → E → 5 → NO-G → 7-EndIf

Task 3

Within a team developing a Java application, clarify who is responsible for white-box/static testing and which tools are used for this purpose.

For this task, you are required to conduct independent research and find examples of tools and frameworks used for testing a Java-based application.

Responsibilities:

- **Developers** – they are the primary ones responsible, as they write the code and perform initial code reviews.
- **QA Engineers/Testers** – they analyze the code to identify bugs and logic errors.
- **Code Reviewers** – team members dedicated to reviewing the code; they also conduct static analysis.
- **Security Testers** – they examine the code for security issues and ensure compliance with security best practices.

Tools and frameworks used:

- **JUnit** – a popular framework for writing and executing unit tests in Java.
- **JMeter** – used for performance testing and load testing.
- **PyTest** – a testing framework for Python (included here only if Python is used alongside Java).
- **Checkstyle** – a tool that verifies if the Java code follows coding standards.
- **Selenium** – used for integration and UI testing in web applications.
- **Fortify Static Code Analyzer** – detects security vulnerabilities through static code analysis.
- **FindBugs** – a static analysis tool that identifies software bugs in Java programs.
- **Mockito** – a mocking framework used in unit and integration testing.
- **Git and Jira** – used for version control and task management.