# Common Logic Errors in Paper 2

This guide by **FutureLogic Education** explains the 12 most common logic mistakes students make in IGCSE Computer Science Paper 2 programming questions — with incorrect and correct code examples for each one.

## 1 Logic Error 1 — Infinite Loops

Students forget to update the loop variable inside the loop, causing it to run forever.

**✗ Incorrect**

```
count = 1
WHILE count <= 5 DO
OUTPUT count
# count never updated!
ENDWHILE
```

**✓ Correct**

```
count = 1
WHILE count <= 5 DO
OUTPUT count
count = count + 1 # fixed
ENDWHILE
```

**EXAM TIP: Every WHILE loop must update its condition variable inside the loop body. Check this first.**

*Teacher note: Ask students to trace the loop manually — they immediately see the counter never changes.*

## 2 Logic Error 2 — Incorrect IF Conditions

Students use = (assignment) instead of == (comparison) inside IF conditions, or use the wrong operator entirely.

| ✗ Incorrect | ✓ Correct |
|---|---|
| `IF score = 50 THEN`<br>`OUTPUT "Pass"`<br>`ENDIF`<br>`# = assigns, does not compare` | `IF score >= 50 THEN`<br>`OUTPUT "Pass"`<br>`ENDIF`<br>`# == or >= compares correctly` |

**EXAM TIP: = assigns a value. == checks equality. Always use == (or >=, <=) inside IF conditions.**

*Teacher note: Show students what happens when = is used — the variable gets overwritten. Visual impact sticks.*

## 3 Logic Error 3 — Using Wrong Variable Names

Students define a variable with one name but reference it later with a different name, causing logic failures.

| ✗ Incorrect | ✓ Correct |
|---|---|
| `totalScore = 0`<br>`FOR i = 1 TO 5 DO`<br>`total = total + marks[i]`<br>`# "total" != "totalScore"`<br>`ENDFOR` | `totalScore = 0`<br>`FOR i = 1 TO 5 DO`<br>`totalScore = totalScore + marks[i]`<br>`# consistent name used`<br>`ENDFOR` |

**EXAM TIP: Use the exact same variable name everywhere. One typo causes silent logic errors that are hard to find.**

*Teacher note: Encourage students to declare all variables at the top and copy-paste names rather than retyping.*

## 4 Logic Error 4 — Incorrect Indentation

Poor indentation places statements outside loops or conditions when they should be inside, changing program logic.

**✗ Incorrect**

```
FOR i = 1 TO 5 DO
total = total + scores[i]
OUTPUT "Total: ", total
# OUTPUT runs 5 times!
ENDFOR
```

**✓ Correct**

```
FOR i = 1 TO 5 DO
total = total + scores[i]
ENDFOR
OUTPUT "Total: ", total
# OUTPUT runs once only
```

**EXAM TIP: Indentation defines what is inside a loop or condition. Always check what runs once vs many times.**

*Teacher note: Trace the code with students — count how many times OUTPUT runs in each version. The difference is immediate.*

## 5 Logic Error 5 — Forgetting Initial Values

Accumulators and counters must be initialised before use. Without an initial value, the result is unpredictable.

**✗ Incorrect**

```
FOR i = 1 TO 10 DO
total = total + scores[i]
# total was never set to 0!
ENDFOR
OUTPUT total
```

**✓ Correct**

```
total = 0 # initialised
FOR i = 1 TO 10 DO
total = total + scores[i]
ENDFOR
OUTPUT total
```

**EXAM TIP: Always initialise counters and totals to 0 before a loop. Declare variables before using them.**

*Teacher note: A missing initialisation is one of the most common trace table errors. Reinforce before every loop exercise.*

## 6 Logic Error 6 — Off-by-One Errors

Loop boundaries are wrong by one — the loop runs one too many or one too few times.

| ✗ Incorrect | ✓ Correct |
| --- | --- |
| `FOR i = 1 TO 9 DO`<br>`OUTPUT scores[i]`<br>`# misses index 10!`<br>`ENDFOR` | `FOR i = 1 TO 10 DO`<br>`OUTPUT scores[i]`<br>`# processes all 10 items`<br>`ENDFOR` |

**EXAM TIP: Count how many times the loop should run and check the boundary values match exactly.**

*Teacher note: Trace tables are the best way to catch off-by-one errors. Insist students trace their loops before finalising.*

## 7 Logic Error 7 — Incorrect Boolean Logic

AND and OR conditions are mixed up, causing the wrong branch to execute.

**✗ Incorrect**

```
IF age > 12 OR age < 18 THEN
OUTPUT "Teenager"
ENDIF
# OR = always true for any age!
```

**✓ Correct**

```
IF age >= 13 AND age <= 18 THEN
OUTPUT "Teenager"
ENDIF
# AND = both must be true
```

**EXAM TIP: AND = both conditions must be true. OR = at least one must be true. Test with boundary values.**

*Teacher note: Ask students: "What age fails the OR version?" Answer: none. That illustrates the error immediately.*

## 8 Logic Error 8 — Forgetting to Validate Input

Programs accept any input without checking it is valid, leading to incorrect results or crashes.

**✗ Incorrect**

```
age = int(input("Enter age: "))
OUTPUT "Your age is ", age
# no check — accepts -5, 999
```

**✓ Correct**

```
age = int(input("Enter age: "))
WHILE age < 0 OR age > 120 DO
age = int(input("Invalid. Re-enter: "))
ENDWHILE
OUTPUT "Your age is ", age
```

**EXAM TIP: Always validate input. Use a WHILE loop to keep asking until the input is within the valid range.**

*Teacher note: Normal, boundary, and invalid data testing is an examiner favourite. Teach validation alongside loops.*

## 9 Logic Error 9 — Wrong Loop Type

Students use a FOR loop when the number of iterations is unknown, or a WHILE loop when it is fixed.

**✗ Incorrect**

```
# Unknown iterations — use WHILE
FOR i = 1 TO 100 DO
answer = input("Try again? ")
IF answer = "no" THEN EXIT
ENDFOR
```

**✓ Correct**

```
# Unknown iterations — WHILE correct
answer = "yes"
WHILE answer <> "no" DO
answer = input("Try again? ")
ENDWHILE
```

**EXAM TIP: FOR = known number of iterations. WHILE = repeat until a condition changes. Choose based on what you know.**

*Teacher note: Ask: "Do we know how many times this runs?" Yes → FOR. No → WHILE. Simple decision rule.*

## 10 Logic Error 10 — Incorrect Totals or Counters

Counters are placed outside the loop or updated incorrectly, giving wrong totals.

**✗ Incorrect**

```
count = 0
FOR i = 1 TO 10 DO
IF scores[i] > 50 THEN
OUTPUT scores[i]
count = count + 1 # outside IF!
ENDFOR
```

**✓ Correct**

```
count = 0
FOR i = 1 TO 10 DO
IF scores[i] > 50 THEN
OUTPUT scores[i]
count = count + 1 # inside IF
ENDFOR
```

**EXAM TIP: Counter updates must be inside the correct block. Indentation determines when they execute.**

*Teacher note: Trace tables catch this immediately. Students who trace their code find this error before submission.*

## 11 Logic Error 11 — Not Testing Programs

Students submit programs without tracing or testing, missing errors that a simple trace table would catch.

**✗ Incorrect**

```
# Student writes code and submits
# without checking output.
# Logic errors go unnoticed.

# No trace = no marks for testing
```

**✓ Correct**

```
# Trace table: i | score | total
# 1 | 45 | 45
# 2 | 72 | 117
# Confirms logic is correct.
# Test: normal, boundary, invalid
```

**EXAM TIP: Always trace your code with at least one set of test data before finalising your answer.**

*Teacher note: Give students a trace table template. Making it a habit during practice means it becomes automatic in the exam.*

## 12 Logic Error 12 — Outputting the Wrong Variable

The calculation is correct but the student outputs the wrong variable — displaying an input value instead of the result.

**✗ Incorrect**

```
total = 0
FOR i = 1 TO 5 DO
total = total + scores[i]
ENDFOR
OUTPUT scores[i] # wrong! outputs last
score
```

**✓ Correct**

```
total = 0
FOR i = 1 TO 5 DO
total = total + scores[i]
ENDFOR
OUTPUT total # correct — outputs result
```

**EXAM TIP: Always check: are you outputting the result variable, not the input or loop variable?**

*Teacher note: A final read-through of OUTPUT statements catches this. Train students to verify every output before submitting.*

## FutureLogic Paper 2 Tips

✔ **Trace your code line-by-line.**
Use a trace table for every loop or condition — it catches most logic errors before submission.

✔ **Test with normal, boundary, and invalid data.**
Normal: typical value. Boundary: edge of valid range. Invalid: outside valid range.

✔ **Use meaningful variable names.**
totalScore is clearer than x. Meaningful names prevent wrong-variable errors.

✔ **Check loop conditions carefully.**
Ask: when does this loop start? When does it stop? How many times does it run?

✔ **Read the question requirements fully.**
Identify exactly what the output should be before writing a single line of code.

## Quick Reference Summary

| # | Error | Fix |
|---|---|---|
| 1 | **Infinite loop** | Update loop variable inside loop |
| 2 | **Wrong IF operator** | Use == for comparison, not = |
| 3 | **Wrong variable name** | Use consistent names throughout |
| 4 | **Bad indentation** | Check what is inside vs outside loops |
| 5 | **No initial value** | Set counters/totals to 0 before loop |
| 6 | **Off-by-one** | Check boundary values carefully |
| 7 | **Wrong Boolean logic** | AND = both true. OR = one true |
| 8 | **No input validation** | Use WHILE loop to reject invalid input |
| 9 | **Wrong loop type** | Known count → FOR. Unknown → WHILE |

| | | |
|---|---|---|
| 10 | **Wrong counter position** | Counter update must be inside correct block |
| 11 | **No testing** | Always trace with a trace table |
| 12 | **Wrong OUTPUT variable** | Check you output the result, not the input |