

Exercise walk-through

1.4

Identifying and Correcting Errors

AP Computer Science Principles

Four kinds of error

Syntax error

breaks language rules
→ will not run

e.g. missing)

Logic error

runs, but gives the
wrong result

e.g. + instead of -

Runtime error

crashes while running

e.g. $\div$ by zero

syntax: won't run · logic: wrong answer · runtime: crash

Syntax error – breaks
the language rules; won't run

Logic error –
runs, but wrong answer

Runtime error –
crashes mid-run (divide by zero)

Overflow error –
value too large for its storage

Name the error, then name the test that catches it

syntax error

breaks the language's rules – caught by the compiler or interpreter before it runs

run-time error

crashes during execution – dividing by zero, indexing past the end

logic error

runs and gives the **wrong answer** – the hardest, because nothing complains

overflow error

a value too large for the bits available

how to catch each

read the message for the first two; **hand-trace** or test with known values for a logic error

This pairing – error type plus a testing strategy that would find it – is a recurring multiple-choice **and** Create-task theme.

Finding & fixing bugs

A **bug** is a mistake; **debugging** is finding & fixing it.

test many inputs,
incl. **edge**
cases

compare with
the **expected**
output

divide & check; re-
test after each fix

A logic error runs with no crash – “it ran” is not “it’s correct”. Only comparing output to what you expected catches it.

Hand-tracing with a trace table

A **trace table** records each variable's value as the program runs – line by line, by hand.

It is how you find a logic error that no crash will ever reveal.

count	total	output
1	5	
2	12	
3	20	20

each variable, at each step

Worked example – a logic error hiding in plain sight

A program should print the average of two numbers, but runs $\text{avg} = a + b / 2$

Order of operations: $/$ runs **before** $+$, so it computes $a + \frac{b}{2}$ – not the average. Test with $a = 4$, $b = 6$: the bug gives $4 + 3 = 7$; the fix $\text{avg} = (a + b) / 2$ gives $\frac{10}{2} = 5$.

Testing with **known inputs** is exactly how you find **and confirm** a logic error – the program never crashed.

Exercises 2.1-2.3

2.1 State the difference between a syntax error and a logic error.

[2]

2.2 State what edge cases are used for in testing.

[1]

2.3 Name one debugging strategy.

[1]

Exercises 2.1-2.3

2.1 State the difference between a syntax error and a logic error. [2]

2.1

- a syntax error breaks the language's rules so the program will not run; a logic error runs but gives the wrong result

2.2 State what edge cases are used for in testing. [1]

2.2

- to test the program at boundaries and unusual values where defects often hide

2.3 Name one debugging strategy. [1]

2.3

- hand-tracing the code, or adding print statements / using a debugger (any one)

Exercises 3.1-3.2

3.1 A program runs but produces the wrong answer. This is a

[1]

☐

A syntax error

B logic error

C overflow error

D case with no error

3.2 Testing a program with unusual or boundary inputs uses

[1]

☐

A edge cases

B comments

C variables

D loops

Exercises 3.1-3.2

3.1 A program runs but produces the wrong answer. This is a

[1]

☐

- A syntax error
- B logic error
- C overflow error
- D case with no error

3.1 B

- logic error.

3.2 Testing a program with unusual or boundary inputs uses

[1]

☐

- A edge cases
- B comments
- C variables
- D loops

3.2 A

- edge cases.

Exercise 3.3 (a)

3.3 A program crashes when it tries to divide by zero.

(a) Name the type of error.

[1]

Exercise 3.3 (a)

3.3 A program crashes when it tries to divide by zero.

(a) Name the type of error.

[1]

(a)

- a runtime error

Exercise 3.3 (b)

3.3 A program crashes when it tries to divide by zero.

(b) Name one way to find the bug.

[1]

Exercise 3.3 (b)

3.3 A program crashes when it tries to divide by zero.

(b) Name one way to find the bug.

[1]

(b)

- hand-trace or add print statements to find where it fails (any one)

Exercise 3.3 (c)

3.3 A program crashes when it tries to divide by zero.

(c) State what you should check after fixing it.

[1]

Exercise 3.3 (c)

3.3 A program crashes when it tries to divide by zero.

(c) State what you should check after fixing it.

[1]

(c)

- that it now produces the intended output for its requirements