Exercise walk-through

5.1 Algorithms, traces and correctness evidence

International Baccalaureate · IB Diploma · Computer Science · SL

INTERNATIONAL BACCALAUREATE · IB DIPLOMA · COMPUTER SCIENCE · SL • EXERCISE SHEET

5.1 Algorithms, traces and correctness evidence

Name: _____ Class: _____ Date: _____ **Total: 10 marks**

KEY WORDS algorithm • boundary test

Objective

Algorithms, traces and correctness evidence. Original classroom practice for IB SL; not an official IB paper.

1 Key knowledge and worked examples

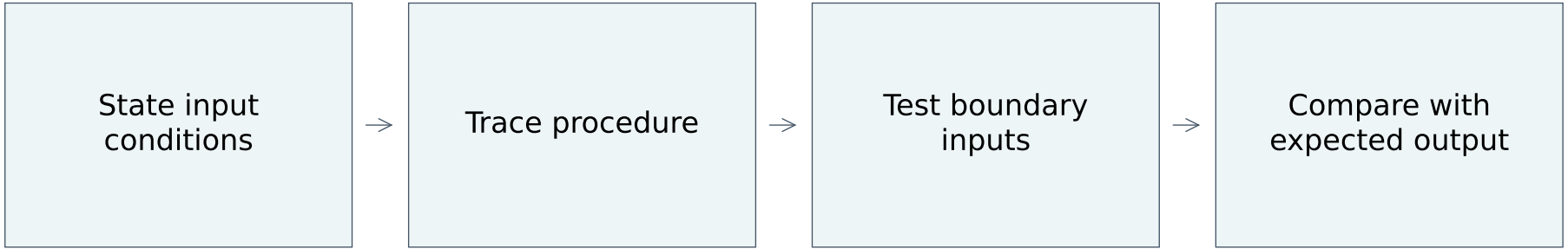
What would explain this observation?

- An **algorithm** can work for one example and fail at a boundary. Testing should be designed from the specification, not only the happy path.

Build the model

- An algorithm is a finite, unambiguous procedure for a task.
- A trace records state changes.
- A loop invariant describes a property preserved by each iteration and helps justify correctness.

- algorithm: A finite procedure solving a stated task.
- **boundary test** : A test at a limit of the allowed input range.



Choose evidence that can test it

- State input conditions and expected outputs.
- Use boundary cases, empty collections where allowed, duplicates and invalid values.
- Distinguish a wrong algorithm from a wrong implementation or an incomplete requirement.

In the lab: Trace a search over a small fictional sorted list. State the indexing convention. For binary search, update bounds so the remaining interval shrinks and reject unsorted input unless sorting is part of the task.

Work from known quantities

- Known: a linear search of an eight-item list can require eight comparisons when the

sought item is last or absent.

- Doubling the list length doubles the worst-case comparison count under this model.
- Binary search reduces the interval by roughly half each step but requires a suitable ordered structure.

Your turn: A linear search scans all 14 items without a match. How many item comparisons occur?

■ Check the conclusion and its limits

- A successful sample test does not prove correctness for all valid inputs.
- Do not import Cambridge-specific pseudocode syntax into an IB course without a course source.

Not true: One successful sample test proves an algorithm correct for every valid input. One passing test shows the algorithm works on that input only. Correctness needs reasoning over all valid inputs or a proof - edge cases such as empty lists and boundary values are exactly where untested code fails.

Key:

- An algorithm is a finite, unambiguous procedure for a task.
- State input conditions and expected outputs.

- Watch for: A successful sample test does not prove correctness for all valid inputs.

Runnable trace and boundary

```
def first_index(values, target):
    for index, value in enumerate(values):
        if value == target:
            return index
    return -1
print(first_index([4, 7, 4], 4))
print(first_index([4, 7, 4], 9))
print(first_index([], 4))
```

Expected output:

```
0
-1
-1
```

The loop tests index 0 first, stops on a match, and returns -1 only after exhausting the list. Duplicates therefore return the first index. An empty list executes no loop body.

Exercises 2.1·2.3

2.1 What must hold before applying ordinary binary search to a list? Choose one answer.

- **A** Every value is different
- **B** The list contains exactly ten items
- **C** The list is ordered under the comparison rule [1]

2.2 A linear search scans all 14 items without a match. How many item comparisons occur? Show the relation, substitution and units. [2]

2.3 Decide whether this claim is true and justify: One passing test shows the algorithm works on that input only. [2]

Exercises 2.1·2.3

2.1 What must hold before applying ordinary binary search to a list? Choose one answer.

- **A** Every value is different
- **B** The list contains exactly ten items
- **C** The list is ordered under the comparison rule [1]

2.1 C Binary search discards half the list by comparing with the middle, which works only if the list is ordered under the comparison rule. Duplicate values are fine, and any length works.

2.2 A linear search scans all 14 items without a match. How many item comparisons occur? Show the relation, substitution and units. [2]

2.2 One equality test per visited item;all14 visited means14 comparisons.

2.3 Decide whether this claim is true and justify: One passing test shows the algorithm works on that input only. [2]

2.3 True. True. One passing test shows the algorithm works on that input only. Correctness needs reasoning over all valid inputs or a proof - edge cases such as empty lists and boundary values are exactly where untested code fails.

Exercises 2.4·3.1

2.4 A finite procedure solving a stated task: write the technical term. [1]

3.1 Trace first-match linear search on [4,7,4] for 4 and 9. Give results and comparison counts, including an empty-list boundary. [4]

Exercises 2.4·3.1

2.4 A finite procedure solving a stated task: write the technical term. [1]

2.4 algorithm. algorithm means A finite procedure solving a stated task.

3.1 Trace first-match linear search on [4,7,4] for 4 and 9. Give results and comparison counts, including an empty-list boundary. [4]

3.1 For 4 return index 0 after one comparison; for 9 return -1 after three. Empty list returns -1 after zero. Use zero-based indices and stop at the first match.