

Computational thinking · supported focus

Supported focus · SL

Supported scope

- Supported SL focus. First assessment 2027 target; official PDF returns 403; older acquired brief is final assessment 2026. Remaining guide, assessment and practical requirements retain their recorded holds.
- State model conditions and distinguish fictional data from measurements.

Computing Algorithms · 1.1

Reasoning

An algorithm can work for one example and fail at a boundary. Testing should be designed from the specification, not only the happy path. Start with a prediction. State the quantities or features you would compare, then decide what evidence could distinguish two explanations.

Computing Algorithms · 2.1

Reasoning

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.

Computing Algorithms · 3.1

Reasoning

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. 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.

Computing Algorithms · 4.1

Reasoning

State the known values and their units. Choose the relation because its assumptions fit this case, then rearrange before substitution. 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.

Computing Algorithms · 4.2

Reasoning

A linear search scans all 14 items without a match. How many item comparisons occur? Use the same sequence: known quantities $\rightarrow$ model $\rightarrow$ relation $\rightarrow$ substitution $\rightarrow$ unit and interpretation.

Computing Algorithms · 5.1

Reasoning

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. Return to the original observation. Explain what the result supports, which conditions it assumes, and one way to test a competing explanation.

Computing Algorithms · 5.2

Reasoning

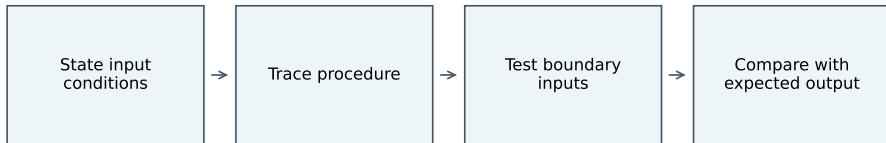
One successful sample test proves an algorithm correct for every valid input. This claim is false: A successful sample test does not prove correctness for all valid inputs. Do not import Cambridge-specific pseudocode syntax into an IB course without a course source.

Computing Algorithms · 5.3

Reasoning

Algorithms, traces and correctness evidence: 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.

Computing Algorithms · diagram



- Original teaching diagram: Algorithms, traces and correctness evidence
- Use the labels and the stated model together.

Computing Algorithms · original transfer

Your turn

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

Computing Algorithms · transfer reasoning

Method and interpretation

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.

Computing Algorithms · executable trace

- Trace the state changes and boundary guard.
- Run the boundary case and compare the printed output.

```
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))
```

Check yourself

- Explain algorithm using the stated model.
- Explain boundary test using the stated model.
- Change one value in the original transfer; explain which assumption still matters.