

9.1 Computing case analysis: evaluate a proposed system

Name: _____ Class: _____ Date: _____

Total: 10 marks

KEY WORDS throughput • acceptance test

Objective

Computing case analysis: evaluate a proposed system. Supported HL concept preparation; original classroom practice, not an official IB paper or assessed learner work.

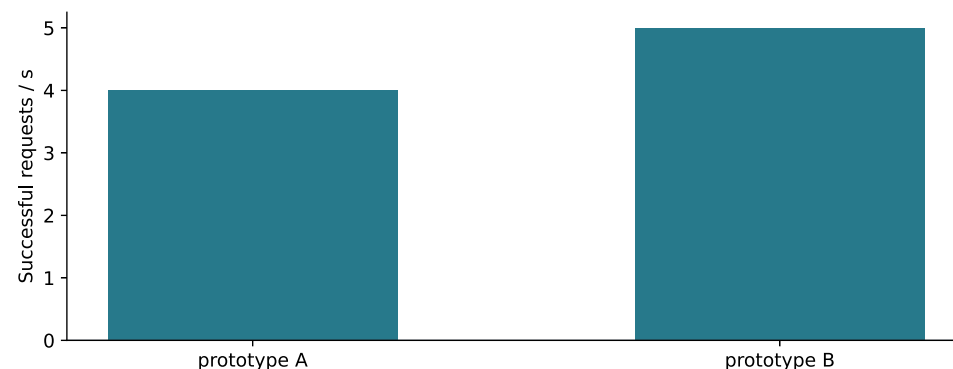
1 Key knowledge and worked examples

■ What would explain this observation?

- A proposed school booking system looks efficient, but its recommendation depends on users, constraints and evidence. A plausible technical term alone is not a justified decision.
- Start with a prediction. State the quantities or features you would compare, then decide what evidence could distinguish two explanations.

■ Build the model

- A case study places computational choices within a stated scenario. Identify stakeholders, requirements, available evidence and constraints before recommending an architecture or algorithm. Distinguish a scenario fact from an assumption and a prediction. Trade-offs can involve performance, reliability, accessibility, privacy, maintenance and cost.
- **throughput** : Completed work per unit time under a stated workload; **acceptance test** : A test of whether a specified user requirement is met.



■ Choose evidence that can test it

- A recommendation should connect a requirement to a mechanism and an observable test. For example, a concurrency problem requires a strategy that prevents conflicting updates, plus tests demonstrating the invariant. A claim about faster response requires comparable workload measurements rather than only a complexity label. Evaluate alternatives under the same stated conditions.
- Use an original school-approved scenario and synthetic data. Build a matrix of claim, supporting scenario evidence, technical explanation, limitation and acceptance test. Mark missing facts explicitly and show how the recommendation would change if an assumption failed. This prepares case-based reasoning without reproducing an unavailable IB assessment case study or inventing its examination rubric.

■ Work from known quantities

- State the known values and their units. Choose the relation because its assumptions fit this case, then rearrange before substitution.
- Known: a fictional prototype processes 120 successful requests during a 30-second observation. Throughput is $120/30=4$ successful requests/s. A second prototype processes 150 in 30 seconds, or 5/s. The second has 25% greater recorded throughput, but no conclusion about latency, failure rate or performance under a larger workload follows without those measurements.

Example:

A fictional system successfully processes 210 requests in 30 s. Calculate throughput. Use the same sequence: known quantities → model → relation → substitution → unit and interpretation.

■ Check the conclusion and its limits

- Throughput is not response time. A strong recommendation states the conditions in which it is expected to work and the evidence that could disconfirm it. Current official CS case-study requirements must be checked against the applicable assessment version.
- Return to the original observation. Explain what the result supports, which conditions it assumes, and one way to test a competing explanation.

Warn:

Higher observed throughput automatically proves lower response time for every request. This claim is false: Throughput is not response time. A strong recommendation states the conditions in which it is expected to work and the evidence that could disconfirm it. Current official CS case-study requirements must be checked against the applicable assessment version.

Key:

Computing case analysis: evaluate a proposed system: A recommendation should connect a requirement to a mechanism and an observable test. For example, a concurrency problem requires a strategy that prevents conflicting updates, plus tests demonstrating the invariant. A claim about faster response requires comparable workload measurements rather than only a complexity label. Evaluate alternatives under the same stated conditions.

■ Runnable trace and boundary

```
for successful, attempted, seconds in [(240, 300, 30), (270, 300, 30)]:
    if seconds <= 0 or attempted <= 0:
        raise ValueError("Positive exposure required")
    throughput = successful / seconds
    failure_fraction = (attempted - successful) / attempted
    print(throughput, failure_fraction)
```

Expected output:

```
8.0 0.2
9.0 0.1
```

Success throughput and failure fraction use different denominators. Zero time or zero attempts cannot support those rates. These fictional summaries do not supply latency, cost or representative-load evidence.

2 Practice

2.1 Which statement best supports a technical recommendation? Choose one answer.

- **A** A mechanism tied to a scenario requirement and a measurable acceptance test
- **B** A popular architecture name without scenario evidence
- **C** A throughput measurement claimed to prove every performance property [1]

2.2 A fictional system successfully processes 210 requests in 30 s. Calculate throughput. Show the relation, substitution and units. [2]

2.3 Decide whether this claim is true and justify: Higher observed throughput automatically proves lower response time for every request. [2]

2.4 Completed work per unit time under a stated workload: write the technical term. [1]

3 Exam-style questions

3.1 Fictional prototypes complete 240/300 and 270/300 requests in 30 s. Compare success throughput and failure fractions. What else is needed for a purchase decision? ^[4]
