2.1 COMPUTATIONAL THINKING · 2.1.4 & 2.1.5

Logical & concurrent — Mark scheme

30 marks · spec 2.1.4–2.1.5

AO key: AO1 = knowledge · AO2 = application · AO3 = reasoned judgements. Credit any valid alternative; do not award the same point twice.

Q	ANSWER	AO	MARKS
1(a)	A point where the program must choose between alternatives / branch. (1)	AO1	1
1(b)	2 decision points, each with its condition (1 for point + 1 for condition, max 4): "is the PIN correct?" — `entered == storedPIN` (2); "is there enough money?" — `balance >= amount` (2). (4)	AO2	4
1(c)	The test (evaluating to True/False) that decides which path is taken. (1)	AO1	1

Q	ANSWER	AO	MARKS
2(a)	Line 02. (1)	AO2	1
2(b)	`spend > 100`. (1)	AO2	1
2(c)	True → line 03 runs (apply the 10% discount) (1); False → line 05 runs instead (full price), the other branch skipped (1). (2)	AO2	2
2(d)	`spend` = 120 → `pay` = 108 (1); `spend` = 80 → `pay` = 80 (1). (2)	AO2	2

Q	ANSWER	AO	MARKS
3(a)	Carrying out more than one task at the same time / interleaving tasks (1); rather than strictly one after another (1). (2)	AO1	2
3(b)	The three downloads can run concurrently (independent of each other) (1); the total must be sequential — it depends on all the data having arrived first (1). (2)	AO2	2
3(c)	Any one: weather forecasting · image/video processing · scientific simulation · large searches. (1)	AO1	1

Q	ANSWER	AO	MARKS
4(a)	1 mark each, max 2: finishes sooner / faster (esp. multi-core) · system stays responsive · handles large divisible problems efficiently. (2)	AO1	2
4(b)	1 mark each, max 2: harder to design/debug · shared-data clashes / race conditions · dependencies can't be parallelised · overhead so speed-up less than proportional. (2)	AO1	2

<table>
<tr><th>Q</th><th>LEVELS-OF-RESPONSE MARK SCHEME (9 MARKS)</th><th>AO</th><th>MARKS</th></tr>
<tr><td>5</td><td>Mark using the levels descriptors below. AO1 (knowledge of concurrency), AO2 (application to the grid model), AO3 (balanced judgement).

<table><tr><th>LEVEL</th><th>DESCRIPTOR</th></tr><tr><td>Level 3 (7–9)</td><td>Thorough, balanced discussion of benefits and trade-offs applied to the grid model, with a justified conclusion. Clear reasoning; accurate terminology.</td></tr><tr><td>Level 2 (4–6)</td><td>Benefits and trade-offs with some development and application; a basic conclusion.</td></tr><tr><td>Level 1 (1–3)</td><td>A few points, largely one-sided, little development.</td></tr><tr><td>0</td><td>Nothing creditworthy.</td></tr></table>
Indicative content (credit any valid point):
• Benefits: many grid cells are independent within a time-step, so they compute concurrently on multiple cores → forecast finishes far sooner; scales with cores.
• Trade-offs: each time-step depends on the previous, so steps stay sequential; coordinating/splitting work adds overhead; harder to design and debug; shared boundary data needs careful handling; speed-up is less than proportional to cores.
• Application: "concurrent within a step, sequential between steps".
• A justified conclusion (e.g. concurrency is worthwhile here because the work divides cleanly, provided dependencies are respected).</td><td>AO1 ×3
AO2 ×3
AO3 ×3</td><td>9</td></tr>
</table>

Total for paper: 30 marks