2.1 COMPUTATIONAL THINKING · 2.1.3

Thinking procedurally — Mark scheme

30 marks · spec 2.1.3(a)–(d)

AO key: AO1 = knowledge · AO2 = application. Accept any valid alternative; sub-procedure names need only be sensible. Do not award the same point twice.

Q	ANSWER	AO	MARKS
1(a)	Breaking a problem into smaller (sub-)components/parts that are easier to solve. (1)	AO1	1
1(b)	1 mark each, max 3, e.g. `loadQuestions` · `askQuestion` · `checkAnswer` · `updateScore` · `showResult`. (3)	AO2	3
1(c)	Any one developed: each part is easier to write/understand (1); can be tested/ reused independently / split among a team (1). (2)	AO1	2

Q	ANSWER	AO	MARKS
2(a)	1 mark each, max 2, e.g. `takePayment` · `confirmBooking` · `issueTickets` · `checkSeatAvailability`. (2)	AO2	2
2(b)	How a problem is decomposed (1); into a hierarchy of sub-tasks/sub-procedures (top problem broken into smaller parts) (1). (2)	AO1	2
2(c)	A named block of code/routine (1); that performs one part of the overall task (1). (2)	AO1	2

Q	ANSWER	AO	MARKS
3(a)	`scanCard` → `scanBook` → `checkLimit` → `recordLoan` → `printReceipt`. All correct (2); mostly correct (1). (2)	AO2	2
3(b)	The loan should only be recorded if the member is under their limit (1); so the check must happen first, otherwise an over-limit loan could be recorded (1). (2)	AO2	2
3(c)	A sensible choice + two parts, e.g. `checkLimit` (1) → `countCurrentLoans` and `compareToLimit` (1). (2)	AO2	2

Q	ANSWER	AO	MARKS
4(a)	1 mark each, max 3: accepting the user's item selection · accepting/counting money inserted · checking enough money · dispensing the drink · giving change. (3)	AO2	3
4(b)	1 mark each, max 2, e.g. `readSelection` · `checkPayment` · `dispenseItem` · `giveChange`. (2)	AO2	2
4(c)	Any one that must precede dispensing, e.g. `checkPayment` (enough money inserted). (1)	AO2	1

Q	ANSWER	AO	MARKS
5(a)	Decomposition = breaking a problem into smaller parts/sub-problems (1); abstraction = removing/hiding irrelevant detail to focus on what matters (1). (2)	AO1	2
5(b)	1 mark each, max 2: easier to write/understand · can be tested/debugged separately · reusable · can be shared among a team. (2)	AO1	2
5(c)	Some steps depend on others / must happen in a set order (1); the solution only works if the steps run in the correct sequence (1). (2)	AO1	2

Total for paper: 30 marks