2.2 PROGRAMMING TECHNIQUES · 2.2.1(B)

Recursion — Mark scheme

30 marks · spec 2.2.1(b)

AO key: AO1 = knowledge · AO2 = application · AO3 = programming. Code answers are indicative — award for correct logic in any consistent language.

Q	ANSWER	AO	MARKS
1(a)	A function/subroutine that calls itself (1); to solve a problem by breaking it into smaller versions of itself (1). (2)	AO1	2
1(b)	The simplest case, solved directly without recursing — the stop condition. (1)	AO1	1
1(c)	The case that calls the function on a smaller input, moving towards the base case. (1)	AO1	1
1(d)	It calls itself forever / never stops (1); the call stack runs out of memory → a stack overflow (crash) (1). (2)	AO1	2

Q	ANSWER	AO	MARKS
2(a)	<table><tr><th>CALL</th><th>RETURNS</th></tr><tr><td>power(2, 3)</td><td>8</td></tr><tr><td>power(2, 2)</td><td>4</td></tr><tr><td>power(2, 1)</td><td>2</td></tr><tr><td>power(2, 0)</td><td>1</td></tr></table> power(2,0)=1 (1); power(2,1)=2 (1); power(2,2)=4 (1); power(2,3)=8 (1). (4)	AO2	4
2(b)	8. (1)	AO2	1
2(c)	Line 03 (the `return 1` when `exp == 0`). (1)	AO2	1

Q	ANSWER	AO	MARKS
3(a)	Each recursive call is pushed onto the stack with its data/return point (1); when the base case is reached the calls are popped and return in reverse order (1). (2)	AO1	2
3(b)	When the call stack runs out of memory (1); usually caused by a missing/ unreachable base case or recursing too deeply (1). (2)	AO1	2
3(c)	Every unfinished call stays on the stack, each taking memory (1); whereas an iterative loop uses a fixed amount of memory (1). (2)	AO1	2

Q	ANSWER	AO	MARKS

4(a)	Indicative solution (award for logic):
	`function power(base, exp)`
	`    result = 1`
	`    for i = 1 to exp`
	`        result = result * base`
	`    next i`
	`    return result`
	`endfunction`
	Initialise `result = 1` (1); loop `exp` times (1); multiply `result` by `base` each pass (1); return `result` (1). (Accept a WHILE loop.) (4)
		AO3	4

4(b)	1 mark each, max 2: uses less/fixed memory (no growing stack) · no risk of stack overflow · usually faster (no call overhead). (2)	AO2	2

Q	ANSWER	AO	MARKS
5(a)	1 mark each, max 2: shorter/more elegant code for naturally recursive problems · closely matches the mathematical/recursive definition · simpler for problems like tree traversal / divide-and-conquer. (2)	AO1	2
5(b)	Any one: tree traversal · divide-and-conquer (e.g. merge/quick sort) · factorial/ Fibonacci · navigating nested/hierarchical structures. (1)	AO1	1
5(c)	Recursion uses stack memory that grows with depth (1); risking a stack overflow / using more memory (1); and is often slower due to call overhead, so iteration is better when performance/memory matters or nesting is deep (1). (3)	AO2	3

Total for paper: 30 marks