Task	Expected response		Additional guidance	Marks available	
1	Software design and development				
1b	Array used to store each weight		Must be used and not simply declared	1	Implementation
	loop 5 times		Loop must contain code	1	
	Input Validation	Input validation — conditional while loop used		1	
		Input validation — correct loop condition	< 0 or > 200	1	
		Input validation — input of weight within loop	Award 1 mark if implemented without input validation loop	1	
		Input validation — error message displayed inside while loop		1	
	Running total inside loop			1	
	selection statements	small and total between 110 and 140	If conditions have a working range but have one repeating error in the conditions, >110 and <140 >330 and <440 >690 and <900 award 2 marks out of 3. Check operators carefully as incorrect use of AND, OR, NOT may invalidate condition	3	
		medium and total between 330 and 440			
		large and total between 690 and 900			
		Structure matches design (nested or else ifs used)		1	
	Calculate average weight			1	
	Display 5 weight inputs within a second loop			1	
	Display total value and the correct stored message from supplied design		No need to display total with additional text to construct a message	1	
	Display rounded average to 1dp		No need to display rounded average with additional text to construct a message	1	

Task	Expected response	Additional guidance	Marks available	
1	**Software design and development**			
1ci	Printed evidence of test run including all inputs and the output message displayed.	Required inputs: 134.23 74.99 25.31 112.33 53.78 medium Required output message: This weight of food is suitable for your medium dog. Output could be used as evidence of inputs.	1	Testing
1cii	1 mark each for: ♦ Test 1 Expected Result - This weight of food is not recommended for the size of dog ♦ Test 2 Inputs – Weights 1 to 5 should total between 110 and 140 size of dog should be small	Expected output could also be message from candidate's code.	2	
1d	Evaluation of the following for: (efficiency) 1 mark each for: ♦ identifying efficient code in own program ♦ justify why it is efficient (robustness) 1 mark: ♦ how robust the program is, including if it copes with unexpected inputs (readability) 1 mark: ♦ readability — comment on one aspect of readability in the candidate's own code	Efficiency examples could include: ♦ use of array ♦ nested ifs ♦ running total inside same loop as inputs ♦ use of a loop Evaluation must contain an element of evaluation rather than simple statements of terms. For example "I have used white space to highlight structures in my program" not "I have used white space". The candidate's code must also show evidence of this for a mark to be awarded.	4	Evaluation