

THEME B: COMPUTATIONAL THINKING · B2.3

Programming constructs — Mark scheme

34 marks · covers B2.3.1–B2.3.4

How to mark. Each creditworthy point is on its own line and ends with a semicolon (;). A line beginning with ... is an *expansion* of the line above it, so the pair is worth the marks shown. A slash (/) separates acceptable alternative wordings. Words in (brackets) are not needed for the mark. Mark positively: credit what the student has got right rather than penalising what is missing, and accept any valid alternative not listed here. Do not award more than the maximum for a part. Where a later part uses a wrong answer from an earlier part correctly, award follow-through marks (FT). HL marks parts that assess higher level only content.

Q	ANSWER	MARKS
1(a) B2.3.1	Award [3] max Award [1] for B before C; [1] for C before A; [1] for A before D. B, C, A, D; The running total must be set to 0 before anything is added to it, all the marks must be added before the average is calculated, and the average must be calculated before it is printed. Award [3] only for the fully correct order.	3
1(b) B2.3.1	Award [2] max Award [1] for the point and [1] for the consequence. total would be reset to 0 after the loop had already added up all the marks; ...(so) the average would be calculated as 0 divided by count, and the program would print 0 instead of the correct average;	2
2(a) B2.3.2	Award [5] max Award [1] for a correct first condition; [1] for using else if / elif rather than separate if statements; [1] for both remaining boundaries correct; [1] for a final else covering everything below 50; [1] for printing the grade in every branch. Accept either language. JAVA <pre> if (mark >= 70) System.out.println("A"); else if (mark >= 60) System.out.println("B"); else if (mark >= 50) System.out.println("C"); else System.out.println("D"); </pre> PYTHON <pre> if mark >= 70: print("A") elif mark >= 60: print("B") elif mark >= 50: print("C") </pre>	5

Q	ANSWER	MARKS
	<pre>else: print("D")</pre> Award [3] max for four separate <code>if</code> statements without <code>else</code>, which would print several grades for one mark. Accept a correct solution that tests in ascending order using <code><</code>, provided the boundaries are right.	
2(b) B2.3.2	Award [1] max B;	1
2(c) B2.3.2	Award [1] max Each condition is only reached when all the earlier ones were false, so testing the highest boundary first is what makes the lower tests correct; testing 50 first would give every mark of 70 the grade C;	1
3(a) B2.3.2	Award [4] max Award [1] for the membership and spend conditions combined with AND; [1] for the Tuesday condition; [1] for joining the two parts with OR; [1] for correct bracketing so that AND is evaluated before OR. JAVA <pre>if ((isMember == true && spend > 50) day.equals("Tuesday")) { // apply discount }</pre> PYTHON <pre>if (isMember and spend > 50) or day == "Tuesday": # apply discount</pre> Accept <code>isMember</code> on its own instead of <code>isMember == true</code>. In Java, strings must be compared with <code>.equals()</code> rather than <code>==</code>, but do not deduct a mark for this alone. Brackets are not strictly required because AND binds more tightly than OR, but award the fourth mark only if the grouping is unambiguous.	4
3(b) B2.3.2	Award [2] max Award [1] for the correct decision and [1] for the justification. Yes, the discount applies; The first part is false because the customer is not a member, but the OR only needs one side to be true, and it is a Tuesday;	2
4(a) B2.3.3	Award [2] max Award up to [1] per difference, [2] max. A counted loop repeats a known number of times, set before the loop starts, whereas a conditional loop repeats until a condition changes; A counted loop is used when the number of repetitions is known in advance, such as processing every item in an array, whereas a conditional loop is used when it is not, such as waiting for valid input;	2
4(b) B2.3.3	Award [4] max Award [1] for reading an input; [1] for a loop that repeats while the value is outside the range; [1] for a correct condition using both boundaries; [1] for re-reading the input inside the loop so it can end. Accept either language.	4

Q	ANSWER	MARKS
	JAVA <pre>Scanner sc = new Scanner(System.in); System.out.print("Enter a number (1-10): "); int n = sc.nextInt(); while (n < 1 n > 10) { System.out.print("Out of range. Try again: "); n = sc.nextInt(); }</pre> PYTHON <pre>n = int(input("Enter a number (1-10): ")) while n < 1 or n > 10: n = int(input("Out of range. Try again: "))</pre> Accept a loop that reads first and tests afterwards (do-while / while True with a break). Award [3] max if the input is not read again inside the loop, since that gives an infinite loop. Accept not <code>(1 <= n <= 10)</code> in Python.	
5(a) B2.3.1 B2.3.3	Award [3] max Award [1] per point up to [3] max. The variable n is never changed inside the loop; The condition <code>n > 0</code> is therefore true on every pass and never becomes false; ...(so) the loop never ends: it is an infinite loop that prints 10 forever; The program will not reach any code after the loop, and must be stopped manually;	3
5(b) B2.3.1	Award [2] max Award [1] for the correct statement and [1] for placing it inside the loop body. JAVA <pre>n = n - 1; // inside the loop, after the println</pre> PYTHON <pre>n = n - 1 # inside the loop, after the print</pre> Accept <code>n - -</code> in Java or <code>n -= 1</code> in Python. Do not award the second mark if the statement is placed outside the loop, which does not fix the fault.	2
6(a) B2.3.4	Award [3] max Award [1] for a correct function header with two parameters; [1] for a correct comparison; [1] for returning a value on every path. Accept either language. JAVA <pre>public static int larger(int a, int b) { if (a > b) return a; else return b; }</pre> PYTHON	3

Q	ANSWER	MARKS
	<pre data-bbox="287 190 1318 376">def larger(a, b): if a > b: return a else: return b</pre> Accept a single-line return using a conditional expression. Award [2] max if the function prints the answer instead of returning it, since the question asks for a return.	
6(b) B2.3.4	Award [2] max Award [1] per benefit, [2] max. Any two. The same code can be reused wherever it is needed rather than being written out again; Each function can be tested on its own, so a fault is easier to locate; The program is easier to read, because each function has a name that says what it does; Different people can work on different functions at the same time; A change only has to be made in one place; Local variables inside a function cannot be altered by the rest of the program;	2

Total for this paper: 34 marks