2.1 Elements of Computational Thinking

OCR H446 · Calculators not allowed · about 125 minutes · Write-in edition

Total Marks

Name: _____ Date: ____ / ____ / _____

Answer ALL questions. Show all your working.

1 Devon is a software engineer working on an air quality sensor for Marlow Council. Each reading is stored as an 8-bit binary value. To save storage space, repeated identical readings are compressed by recording a value once together with a count of how many times it repeats, an application of pattern recognition in computational thinking.

(a) Convert the denary value 173 to an 8-bit binary number. Show your working. **(2)**

(b) Convert your answer to part (a) into hexadecimal. **(2)**

(c) State the name of the pattern recognition technique described above, in which a repeated value is stored once together with a count of how many times it occurs, and state one benefit of using it. **(2)**

(d) A second sensor reading is recorded as FE in hexadecimal. State this value in binary and in denary. **(2)**

(Total for Question 1 is 8 marks)

2 Callum is writing the control program for a vending machine. The machine should display a prompt unless a coin has been inserted AND the selection button has been pressed. His first version of the condition, using two Boolean variables hasCoin and buttonPressed, is:

```
IF NOT (hasCoin AND buttonPressed) THEN
  OUTPUT "Please insert a coin and press the button"
ENDIF
```

(a) Complete a truth table for the expression NOT (hasCoin AND buttonPressed) for all four combinations of hasCoin and buttonPressed. **(3)**

(b) Show that NOT (hasCoin AND buttonPressed) is logically equivalent to (NOT hasCoin) OR (NOT buttonPressed). **(3)**

(c) Rewrite the IF statement above so that it uses the simplified logical expression from part (b) instead, using OCR Exam Reference Language. **(3)**

(Total for Question 2 is 9 marks)

3 Kelvin Grove Hospital models patient waiting times at its walk-in triage desk to test how changes might affect the department, without needing to trial them on real patients (performance modelling). One triage nurse (a single server) sees patients strictly in the order they arrive. Five patients arrive at times (in minutes after opening) of 0, 4, 5, 9 and 15, and require treatment (service) times of 6, 3, 5, 4 and 2 minutes respectively. The algorithm below simulates the queue:

```
arrival[1] <- 0
arrival[2] <- 4
arrival[3] <- 5
arrival[4] <- 9
arrival[5] <- 15
service[1] <- 6
service[2] <- 3
service[3] <- 5
service[4] <- 4
service[5] <- 2
finishPrev <- 0
totalWait <- 0
FOR i <- 1 TO 5
  IF arrival[i] > finishPrev THEN
    start[i] <- arrival[i]
  ELSE
    start[i] <- finishPrev
  ENDIF
  finish[i] <- start[i] + service[i]
  wait[i] <- start[i] - arrival[i]
  finishPrev <- finish[i]
  totalWait <- totalWait + wait[i]
NEXT i
```

(a) State what is meant by performance modelling as a computational thinking technique, and give one reason it is used instead of trialling changes on the real system. **(2)**

(b) Complete a trace table showing the values of start[i], finish[i] and wait[i] for i = 1 to 5, and the final value of totalWait, when the algorithm above is executed. **(6)**

(c) Calculate the mean waiting time per patient across this simulation, giving your answer to 1 decimal place. **(3)**

(d) Explain why simulating a much larger number of patients would improve the reliability of this performance model's predictions, and suggest one limitation that would remain even with a much larger simulation. **(3)**

(Total for Question 3 is 14 marks)

4 Freya analyses monthly visitor numbers for a council website: Jan 1200, Feb 1350, Mar 3100, Apr 1400, May 1450, Jun 1500.

(a) State two reasons why visualising this data (e.g. as a line graph) can make patterns easier to identify than examining the raw numbers alone. **(2)**

(b) Identify, from the data given, the month that is an outlier compared with the general trend, and suggest a possible real-world cause. **(2)**

(c) State one type of visualisation, other than a line graph, that would be more appropriate for showing the proportion of the total annual visitors contributed by each month, and justify your choice. **(2)**

(d) Explain one limitation of relying on visualisation alone, rather than combining it with other computational thinking techniques such as data mining, when analysing very large datasets. **(2)**

(Total for Question 4 is 8 marks)

5 Mr Whitfield stores the exam scores for five students in an array called scores, with indices 0 to 4. The initial contents are scores = [67, 45, 89, 23, 71]. He sorts them using the following algorithm:

```
FOR i <- 0 TO 3
  FOR j <- 0 TO 3 - i
    IF scores[j] > scores[j + 1] THEN
      temp <- scores[j]
      scores[j] <- scores[j + 1]
      scores[j + 1] <- temp
    ENDIF
  NEXT j
NEXT i
```

(a) Complete a trace table showing the contents of the array scores after each comparison (each value of j) during the first pass of the outer loop, i.e. when i = 0. **(4)**

(b) State how many comparisons of the IF condition take place during this first pass (i = 0), for this 5 item list. **(2)**

(c) State an expression, in terms of n, for the total number of comparisons of the IF condition made in the worst case when sorting a list of n items using this algorithm. **(2)**

(d) The inner loop bound (3 - i) means that, as i increases, fewer comparisons are made in each successive pass. Explain why this is a valid optimisation of the algorithm. **(2)**

(Total for Question 5 is 10 marks)

6 Elodie is optimising a program that calculates Fibonacci numbers, where Fibonacci(0) = 0, Fibonacci(1) = 1, and Fibonacci(n) = Fibonacci(n-1) + Fibonacci(n-2) for n ≥ 2. She notices that a simple recursive version recalculates the same values many times over, so she decides to store (cache) results as they are calculated for reuse later, an example of the 'thinking ahead' computational thinking technique.

(a) Define what is meant by memoisation, and state which computational thinking technique it is an example of. **(2)**

(b) The bottom-up algorithm below calculates Fibonacci(n) by filling an array called cache: **(3)**

```
FUNCTION Fibonacci(n)
  DECLARE cache : ARRAY[0:n] OF INTEGER
  cache[0] <- 0
  cache[1] <- 1
  FOR i <- 2 TO n
    cache[i] <- cache[i - 1] + cache[i - 2]
  NEXT i
  RETURN cache[n]
ENDFUNCTION
```

Complete a trace table showing the value stored in cache[i] for each value of i from 2 to 6, when Fibonacci(6) is called.

(c) Rewrite Fibonacci as a recursive function, in OCR Exam Reference Language, that uses memoisation to avoid recomputing values. You should use a global array called cache, with every element initially set to -1 to show that no value has yet been calculated. **(5)**

(d) State one other situation in computing, other than calculating Fibonacci numbers, where caching improves performance, and explain why it helps. **(2)**

(Total for Question 6 is 12 marks)

7 Priya drives a delivery van and must visit customers at four locations, A, B, C and D, before returning to the depot P. The table below shows the distance, in km, between each pair of locations:

P-A=4, P-B=9, P-C=7, P-D=6, A-B=5, A-C=8, A-D=3, B-C=2, B-D=10, C-D=6.

Because checking every possible route becomes impractical as the number of locations grows, Priya's route-planning app uses the nearest-neighbour heuristic: starting at the depot, always travel next to the closest location not yet visited.

Distance matrix (km)

	P	A	B	C	D
P	-	4	9	7	6
A	4	-	5	8	3
B	9	5	-	2	10
C	7	8	2	-	6
D	6	3	10	6	-

(a) State why an exact algorithm that checks every possible route becomes impractical as the number of delivery locations increases. **(2)**

(b) Apply the nearest-neighbour heuristic, starting and ending at the depot P, to find a route visiting A, B, C and D. State the order in which locations are visited and calculate the total distance travelled. **(4)**

(c) Show that the route P-A-B-C-D-P is shorter than the nearest-neighbour route found in part (b), and explain what this shows about the nearest-neighbour heuristic. **(3)**

(d) Give one advantage and one disadvantage of using a heuristic approach, such as nearest-neighbour, rather than an exact algorithm for this kind of real-world routing problem. **(2)**

(Total for Question 7 is 11 marks)

8 Kestrel Stores is a supermarket chain that analyses transaction data collected through its loyalty card scheme, searching for patterns in what customers buy in order to improve stock control and target promotions.

(a) Define what is meant by data mining. **(2)**

(b) Big data is often characterised using several 'V' properties. Other than Volume, identify and briefly explain two of these properties, applying each one to Kestrel Stores' transaction data. **(4)**

(Total for Question 8 is 6 marks)

9 Kestrel Stores (see the previous question) is considering extending its data mining to build detailed individual customer profiles that could predict sensitive health conditions, such as pregnancy or diabetes, purely from patterns in a customer's purchases, without asking the customer directly. Discuss the ethical issues raised by this proposal, and evaluate whether Kestrel Stores would be justified in using data mining in this way.

(Total for Question 9 is 6 marks)

10 Jayden is developing a funds transfer feature for Nexbank's mobile app. Two processes, ProcessA and ProcessB, may run at the same time and both need exclusive access to a shared balanceRecord and a shared transactionLog to complete a transfer:

```
ProcessA:
LOCK balanceRecord
LOCK transactionLog
// transfer funds
UNLOCK transactionLog
UNLOCK balanceRecord
ProcessB:
LOCK transactionLog
LOCK balanceRecord
// transfer funds
UNLOCK balanceRecord
UNLOCK transactionLog
```

(a) Define what is meant by 'thinking concurrently' as a computational thinking technique. **(2)**

(b) Explain, with reference to the pseudocode above, how a deadlock could occur if ProcessA and ProcessB run at the same time. **(3)**

(c) Suggest a change to the order of the LOCK statements in ProcessB that would prevent this deadlock, and explain why your change works. **(3)**

(d) State one difference between a deadlock and a race condition. **(2)**

(Total for Question 10 is 10 marks)

11 TransitLink is a logistics company whose route-planning software must decide the delivery order for up to 200 stops per van each day. Discuss how a combination of computational thinking techniques, decomposition, abstraction, heuristics and performance modelling, could be applied by TransitLink's developers to produce a workable, efficient solution, and evaluate the trade-offs involved in using approximate (heuristic) methods rather than exact algorithms for a problem of this scale.

(Total for Question 11 is 9 marks)

Mark scheme · 2.1 Elements of Computational Thinking

Question 1

(a) correct method, e.g. splitting 173 into place values 128+32+8+4+1 (oe) **M1**

(a) cao 10101101 **A1**

(a) Answer: 10101101

(b) splits the byte into two 4-bit nibbles, 1010 and 1101 (oe) **M1**

(b) cao AD **A1**

(b) Answer: AD

(c) names run-length encoding (RLE) **B1**

(c) states a valid benefit, e.g. reduces the amount of storage space/data volume needed (oe) **B1**

(c) Answer: Run-length encoding (RLE); it reduces the amount of storage space needed to hold the data.

(d) cao binary 11111110 **B1**

(d) cao denary 254 **B1**

(d) Answer: Binary: 11111110, Denary: 254

Question 2

(a) correct hasCoin AND buttonPressed column for all 4 rows (T, F, F, F) **B1**

(a) correct NOT(...) column for all 4 rows (F, T, T, T) **B1**

(a) all four rows fully correct and consistently laid out **B1**

(a) Answer: hasCoin=T,buttonPressed=T: AND=T, NOT=F | T,F: AND=F, NOT=T | F,T: AND=F, NOT=T | F,F: AND=F, NOT=T

(b) states De Morgan's law: NOT (P AND Q) = (NOT P) OR (NOT Q) **M1**

(b) correctly substitutes P = hasCoin and Q = buttonPressed **M1**

(b) cso - concludes the two expressions are equivalent, consistent with the truth table in part (a) **A1**

(b) Answer: NOT (hasCoin AND buttonPressed) = (NOT hasCoin) OR (NOT buttonPressed), by De Morgan's law.

(c) any correct logic: replaces the condition with (NOT hasCoin) OR (NOT buttonPressed) **M1**

(c) correct placement of NOT before each variable **M1**

(c) structure of IF...THEN...ENDIF and the OUTPUT statement preserved unchanged **A1**

(c) Answer:

```
IF (NOT hasCoin) OR (NOT buttonPressed) THEN
  OUTPUT "Please insert a coin and press the button"
ENDIF
```

Question 3

(a) identifies that performance modelling uses a simplified model/simulation to predict how a system will behave (oe) **B1**

(a) valid reason, e.g. cheaper/safer/faster than making changes to and testing a real system, or avoids risk to real patients (oe) **B1**

(a) Answer: Performance modelling uses a simplified model or simulation to predict how a system will behave under different conditions. It is used because trialling changes on the real hospital system could be slower, more expensive, or put real patients at risk.

(b) i=1: start=0, finish=6, wait=0 **B1**

(b) i=2: start=6, finish=9, wait=2 **B1**

(b) i=3: start=9, finish=14, wait=4 **B1**

(b) i=4: start=14, finish=18, wait=5 **B1**

(b) i=5: start=18, finish=20, wait=3 **B1**

(b) cao totalWait=14 **B1**

(b) Answer: i=1: start=0,finish=6,wait=0. i=2: start=6,finish=9,wait=2. i=3: start=9,finish=14,wait=4. i=4: start=14,finish=18,wait=5. i=5: start=18,finish=20,wait=3. totalWait=14.

(c) uses totalWait = 14 from part (b) **M1**

(c) divides by the number of patients, 5 **M1**

(c) cao 2.8 minutes **A1**

(c) Answer: 2.8 minutes

(d) explains that a larger sample smooths out random variation and is more representative of typical patient arrival/service patterns (oe) **B1**

(d) identifies a specific remaining limitation, e.g. the model assumes strict FIFO order and does not account for priority/emergency patients who would be seen out of turn in reality (oe) **B1**

(d) further valid development of the limitation or its consequence, e.g. this means the model may still underestimate waiting times for lower priority patients on a busy day (oe, ft) **B1**

(d) Answer: A larger simulation averages out random variation in arrival and service times, giving a more reliable, representative prediction. However, it would still assume strict first-come-first-served order and would not model priority/emergency patients being seen out of turn, so it could still underestimate waits for some patients in reality.

Question 4

(a) identifies that people can recognise visual patterns/trends more quickly than by reading numeric tables (oe) **B1**

(a) identifies that anomalies or outliers stand out immediately in a chart (oe) **B1**

(a) Answer: A visual chart lets patterns and trends be spotted at a glance, more quickly than scanning a table of numbers; anomalies or outliers (unusually high or low points) stand out immediately.

(b) identifies March (3100) as the outlier **B1**

(b) plausible real-world cause, e.g. a one-off marketing campaign or local news event drove extra visits that month (oe) **B1**

(b) Answer: March (3100 visitors) is the outlier; a possible cause is a one-off marketing campaign or news story that temporarily increased visits.

(c) names an appropriate chart, e.g. a pie chart or a stacked bar chart **B1**

(c) valid justification, e.g. it clearly shows how each month's share compares to the whole/to the other months (part-to-whole relationship) (oe) **B1**

(c) Answer: A pie chart, because it clearly shows how each month's visitor total compares as a proportion of the whole year, rather than showing change over time.

(d) identifies that a chart with a very large number of data points can become cluttered/unreadable, or that the choice of scale can mislead the viewer (oe) **B1**

(d) identifies that data mining can uncover subtle patterns/relationships in the data that are not visible to the human eye in a chart (oe) **B1**

(d) Answer: With very large datasets, a chart can become too cluttered to read clearly, and misleading scales can distort the apparent pattern. Data mining can uncover subtle patterns or relationships in large datasets that would not be visible simply by looking at a chart.

Question 5

(a) j=0: correctly swaps to give [45, 67, 89, 23, 71] **B1**

(a) j=1: no swap, array unchanged [45, 67, 89, 23, 71] **B1**

(a) j=2: correctly swaps to give [45, 67, 23, 89, 71] **B1**

(a) j=3: correctly swaps to give [45, 67, 23, 71, 89] **B1**

(a) Answer: After i=0: [45, 67, 23, 71, 89]

(b) cao 4 **B1**

(b) correct reason, e.g. j takes the 4 values 0, 1, 2 and 3 so the IF condition is tested 4 times (oe, ft) **B1**

(b) Answer: 4 comparisons

(c) recognises the pattern that pass i makes (n-1-i) comparisons, summed over i=0 to n-2 (oe) **M1**

(c) cao n(n-1)/2 (oe equivalent forms accepted) **A1**

(c) Answer: n(n-1)/2

(d) recognises that after each pass, the largest remaining unsorted value has been moved into its correct position at the end of the unsorted section (oe) **B1**

(d) recognises that this position never needs to be compared again, so reducing the range of j reduces the total number of comparisons/improves efficiency without affecting correctness (oe) **B1**

(d) Answer: Each pass guarantees the largest unsorted value bubbles into its correct final position, so that position can be excluded from all later passes, reducing the total comparisons made without changing the result.

Question 6

(a) defines memoisation as storing/caching the results of expensive or repeated calculations so they can be reused instead of recalculated (oe) **B1**

(a) identifies 'thinking ahead' as the computational thinking technique **B1**

(a) Answer: Memoisation is the storing of the results of calculations so that, if the same calculation is needed again, the stored result can be reused instead of being recalculated. It is an example of thinking ahead.

(b) cache[2]=1 and cache[3]=2 both correct **B1**

(b) cache[4]=3 and cache[5]=5 both correct **B1**

(b) cao cache[6]=8 **B1**

(b) Answer: cache[2]=1, cache[3]=2, cache[4]=3, cache[5]=5, cache[6]=8

(c) any correct logic: cache array declared/initialised so that every element holds -1 before the function is first called **M1**

(c) correct base case: IF n ≤ 1 THEN RETURN n (oe) **M1**

(c) correct check: IF cache[n] <> -1 THEN RETURN cache[n] (or equivalent test that a value has already been stored) **M1**

(c) correct recursive calculation cache[n] <- Fibonacci(n - 1) + Fibonacci(n - 2) when no cached value exists **M1**

(c) correct RETURN cache[n] after the value has been stored; ignore minor syntax errors throughout **M1**

(c) Answer:

```
FOR i <- 0 TO n
  cache[i] <- -1
NEXT i

FUNCTION Fibonacci(n)
  IF n <= 1 THEN
    RETURN n
  ENDIF
  IF cache[n] <> -1 THEN
    RETURN cache[n]
  ENDIF
  cache[n] <- Fibonacci(n - 1) + Fibonacci(n - 2)
  RETURN cache[n]
ENDFUNCTION
```

(d) valid example, e.g. a web browser caching images/pages, or a CPU cache storing recently used memory values (oe) **B1**

(d) valid reason linked to the example, e.g. avoids the slower process of re-downloading/re-fetching the data from its original, slower source, reducing loading time/improving performance (oe, ft) **B1**

(d) Answer: e.g. a web browser caches images so that revisiting a page does not require re-downloading them, reducing loading time, because reading from the local cache is much faster than re-fetching data across a network.

Question 7

(a) identifies that the number of possible routes grows factorially as locations are added (oe) **B1**

(a) identifies that this makes checking every route computationally infeasible/take too long for even a moderate number of locations (oe) **B1**

(a) Answer: The number of possible routes grows factorially as more locations are added, so checking every route becomes computationally infeasible (takes too long) even for a moderate number of locations.

(b) correct method: at each step, selects the nearest unvisited location from the current location **M1**

(b) correct order of visits P-A-D-C-B-P **A1**

(b) correct individual distances used (4, 3, 6, 2, 9) **A1**

(b) cao total = 24 km **A1**

(b) Answer: P - A - D - C - B - P, total distance = 24 km

(c) correctly sums the distances for P-A-B-C-D-P: 4+5+2+6+6 **M1**

(c) cso 23 km, correctly identified as less than the 24 km found in part (b) **A1**

(c) explains that this shows the nearest-neighbour heuristic does not guarantee finding the optimal (shortest) route, only a reasonable one found quickly (oe) **B1**

(c) Answer: P-A-B-C-D-P = 4+5+2+6+6 = 23 km, which is less than the 24 km nearest-neighbour route. This shows that the nearest-neighbour heuristic does not always find the optimal route.

(d) valid advantage, e.g. produces a usable route very quickly, even for large numbers of locations, unlike an exact algorithm (oe) **B1**

(d) valid disadvantage, e.g. does not guarantee the optimal (shortest) route will be found, as shown in part (c) (oe) **B1**

(d) Answer: Advantage: it finds a workable route very quickly, even as the number of delivery locations grows. Disadvantage: it does not guarantee the shortest possible route will be found.

Question 8

(a) identifies that data mining involves analysing/searching through large volumes of data (oe) **B1**

(a) identifies that its purpose is to discover previously unknown patterns, trends or relationships within the data (oe) **B1**

(a) Answer: Data mining is the process of analysing large volumes of data using computational techniques in order to discover previously unknown patterns, trends or relationships.

(b) names a valid V, e.g. Velocity **B1**

(b) applies it to the scenario, e.g. transaction data is generated continuously and rapidly at checkouts across many stores and must be processed at speed (oe) **B1**

(b) names a second valid V, e.g. Variety **B1**

(b) applies it to the scenario, e.g. the data includes different formats/types, such as structured till receipts alongside unstructured customer feedback (oe) **B1**

(b) Answer: Velocity: transaction data is generated continuously and at high speed across all stores, and must be processed quickly. Variety: the data comes in different formats, e.g. structured purchase records alongside unstructured customer service notes.

Question 9

Level 1 Identifies a generic or one-sided ethical issue (e.g. privacy) with little or no reference to the Kestrel Stores scenario, and minimal evaluation. **1-2**

Level 2 Identifies more than one relevant ethical issue (e.g. consent, sensitive data, transparency) with some application to the scenario, and attempts an evaluation, though it may be unbalanced or reach no clear conclusion. **3-4**

Level 3 Sustained, balanced discussion of ethical issues (e.g. informed consent, sensitive/health data, transparency, potential for harm) clearly applied to the Kestrel Stores scenario, weighing benefits against risks and reaching a justified evaluative conclusion. **5-6**

Indicative content:

- Inferring sensitive health information (e.g. pregnancy, diabetes) from purchasing patterns without asking the customer directly raises a privacy/consent issue, since customers are unlikely to expect this level of profiling from a loyalty card.
- Customers may not understand, or have agreed to, this specific use of their data (lack of transparency/informed consent), even if it is technically permitted under the loyalty scheme's terms and conditions.
- Health-related information is a special category of sensitive data; using it to target individuals could cause distress or harm if profiling is inaccurate, or if it reveals a condition to others (e.g. via a mailed offer).
- Counter-argument: used responsibly, in aggregate/anonymised form, such analysis can improve stock control and offer genuinely useful, relevant promotions to customers, and is common commercial practice.
- A justified evaluation should weigh the business/customer benefits against the risk of covert profiling of sensitive characteristics without explicit consent, and may conclude that safeguards such as anonymisation, clear opt-in consent, and transparency would be needed for this to be justified.

Question 10

(a) identifies that thinking concurrently involves considering tasks/processes that may run at the same time (oe) **B1**

(a) identifies that it involves designing an algorithm/system so that these simultaneous tasks interact correctly, e.g. share resources safely (oe) **B1**

(a) Answer: Thinking concurrently means designing an algorithm or system with the understanding that multiple tasks or processes may be running/executing at the same time, and ensuring they interact correctly, for example when sharing resources.

(b) identifies that ProcessA locks balanceRecord while ProcessB locks transactionLog at around the same time **M1**

(b) identifies that ProcessA then waits for transactionLog (held by ProcessB), while ProcessB waits for balanceRecord (held by ProcessA) **M1**

(b) cao concludes that neither process can proceed, so a deadlock (circular wait) occurs **A1**

(b) Answer: ProcessA locks balanceRecord while ProcessB locks transactionLog at almost the same time. ProcessA then waits to lock transactionLog, which ProcessB is holding, while ProcessB waits to lock balanceRecord, which ProcessA is holding. Neither process can proceed, so a deadlock occurs.

(c) correctly suggests reordering ProcessB to LOCK balanceRecord before LOCK transactionLog (i.e. the same order as ProcessA) **B1**

(c) explains that both processes now request the shared resources in the same order, so whichever process locks balanceRecord first will be able to acquire transactionLog before the other process can request it (oe) **B1**

(c) further valid development, e.g. identifies this as an example of the general deadlock-prevention technique of enforcing a consistent lock ordering/hierarchy (oe) **B1**

(c) Answer: Reorder ProcessB to LOCK balanceRecord then LOCK transactionLog, the same order as ProcessA. Because both processes now request the two resources in the same order, one process will always obtain both locks in turn without the other holding one it needs, removing the circular wait. This is a standard deadlock-prevention technique: enforcing a consistent lock ordering.

(d) correctly describes deadlock, e.g. two or more processes are permanently blocked, each waiting for a resource held by another (oe) **B1**

(d) correctly describes a race condition, e.g. processes do complete, but the final result depends unpredictably on the timing/order in which they access shared data, and may be incorrect/inconsistent (oe) **B1**

(d) Answer: In a deadlock, two or more processes are permanently blocked, each waiting for a resource held by the other, so none can proceed. In a race condition, the processes do complete, but the outcome depends unpredictably on the exact timing/order of access to shared data, which can produce an incorrect or inconsistent result.

Question 11

Level 1 Limited or generic discussion, largely descriptive rather than applied to the TransitLink scenario; little or no evaluation of the heuristic versus exact trade-off; technical terminology used inconsistently. **1-3**

Level 2 Reasonable discussion of at least two computational thinking techniques, applied with some success to the scenario; some evaluation of the heuristic versus exact trade-off, though it may be one-sided or underdeveloped; mostly appropriate terminology. **4-6**

Level 3 Detailed, coherent discussion of at least three computational thinking techniques, accurately applied to the TransitLink scenario; balanced evaluation of the trade-offs between heuristic and exact approaches, reaching a justified conclusion; technical terminology used accurately throughout. **7-9**

Indicative content:

- Decomposition: the overall route-planning problem can be broken down into smaller sub-problems, e.g. first grouping stops by depot/geographic zone, then deciding the delivery order within each zone.
- Abstraction: the algorithm can ignore irrelevant detail (e.g. individual driver preferences, exact parcel contents) while retaining the essential detail needed to plan a route (e.g. delivery time windows, road distances/times between stops).
- Heuristics: an exact algorithm checking every possible order of up to 200 stops is computationally infeasible, since the number of possible routes grows factorially; a heuristic (e.g. nearest-neighbour, or an improvement heuristic such as 2-opt) can produce a good, workable route in a practical amount of time.
- Performance modelling: a simulation of proposed routes (e.g. under varying traffic conditions or van breakdowns) can be used to test how well a route plan is likely to perform in practice before it is used operationally, without the cost/risk of testing on real deliveries.
- Evaluation of trade-off: heuristic methods are fast and scale well to large numbers of stops, but do not guarantee an optimal (shortest/cheapest) route; exact algorithms guarantee optimality but become impractical well before 200 stops are reached, so a combination of techniques (decomposition to reduce problem size, then heuristics within each sub-problem, refined and tested by performance modelling) is an appropriate, justified real-world solution given the operational need for near real-time route generation.