

Quiz - Week 1

Module 1: Computational Thinking and Algorithm Design

Questions only

Course home

Choose one answer for Questions 1–11. For Questions 12 and 13, complete all the drawing, table, and writing tasks specified in each question.

Name _____ Student number _____

Question 1) Which addition makes the route request complete?

A classmate receives only the route table and this request:

Choose exactly one route with the shortest listed travel time.

Table 1: Routes available for the journey to school.

Route	Listed travel time (minutes)
A	18
B	14
C	14

Two routes share the shortest time. Which addition makes the request produce one predictable result, using only the table and no unstated decision rule?

- A) If several routes share the shortest time, choose all those routes.
- B) If several routes share the shortest time, choose the one with the fewest traffic lights.
- C) If several routes share the shortest time, choose the route whose label comes first alphabetically.
- D) If several routes share the shortest time, choose the one with the minimum listed travel time.

Question 2) Which decomposition completes the presentation task?

A student must prepare a five-slide presentation using three supplied articles. The slides must answer two given questions and name at least one supporting article for each answer. The student must submit the finished slides as one PDF file by 16:00 on Friday. Here, the PDF is an exported copy of the slides.

A *decomposition* divides the task into smaller parts. The result of one part may supply information needed for a later part. Which plan gives connected parts that produce and submit the required final result?

- A)** Extract relevant facts and article names; draft both answers with their sources; create and check the five slides from those drafts; export the checked slides as one PDF and submit it by the deadline.
- B)** Extract relevant facts and article names; create five slides with empty answer sections; draft both answers in a separate file; export the unchanged slides as one PDF and submit it by the deadline.
- C)** Extract the three article titles and author names; draft a summary of those titles and authors; create and check five slides containing that summary; export the slides as one PDF and submit it by the deadline.
- D)** Extract relevant facts and article names; draft both answers with their sources; create five slides and export one PDF; correct missing content in the slides and submit the earlier PDF by the deadline.

Question 3) Which abstraction supports walking-route planning?

You want to find the shortest walking route from home to a sports center. *Shortest* means the smallest total distance, measured in meters. Each street section joins two named places, such as intersections, home, or the sports center. Some sections are closed to pedestrians. Every open section can be walked in either direction.

Which representation keeps enough information to find and compare permitted routes? Each option lists all the information you would have available.

- A)** The named places; a connection between every pair of places; the straight-line distance between each pair.
- B)** The named places; the pairs joined by open pedestrian street sections; the estimated walking time of each section.
- C)** The named places; the pairs joined by any street section, with no record of closures; the length of each section.
- D)** The named places; the pairs joined by open pedestrian street sections; the length in meters of each section.

Question 4) Which set of decision rules covers every case?

A room-entry decision depends on two conditions: *Has a booking?* and *Has a student card?* Each condition can be Yes or No, and all combinations are possible.

You are preparing a decision table with one rule for each combination. In the pairs below, the first entry answers *Has a booking?* and the second answers *Has a student card?* For example, (Yes, No) means a booking and no student card.

Which set includes every possible combination exactly once?

- A) (Yes, Yes), (Yes, No), (No, Yes), (Yes, Yes).
- B) (Yes, Yes), (Yes, No), (No, No), (No, No).
- C) (Yes, Yes), (Yes, No), (No, Yes), (No, No).
- D) (Yes, Yes), (No, Yes), (No, No), (No, Yes).

Question 5) Which decision-table rule matches the situation?

A shop sends a parcel only when its payment is confirmed and its delivery address is valid. Otherwise, the shop keeps the parcel. The decision table records that policy. Each rule is one column.

In the action rows, **X** means *perform the action* and **-** means *do not perform the action*.

Table 2: Complete parcel-sending decision table.

Conditions	Rule 1	Rule 2	Rule 3	Rule 4
Payment confirmed?	Yes	Yes	No	No
Delivery address valid?	Yes	No	Yes	No
Actions				
Send the parcel	X	-	-	-
Keep the parcel	-	X	X	X

For one parcel, the shop has **not confirmed payment** and has checked that the **delivery address is valid**. Which rule matches both conditions, and what action does it require?

- A) Rule 1 applies, so send the parcel.
- B) Rule 2 applies, so keep the parcel.
- C) Rule 3 applies, so keep the parcel.
- D) Rule 4 applies, so keep the parcel.

Question 6) Which change repairs the damaged-item counter?

An *algorithm* is a step-by-step procedure for carrying out a task. The algorithm below should count all damaged items in a finite list. Every item's Damage Status is `Yes` or `No`. For an empty list, the required count is `0`.

The draft below is incorrect. For the statuses `Yes`, `No`, `Yes`, it outputs `1` although the required count is `2`.

The steps use *pseudocode*, a way to describe instructions with words and structured lines. `Input` names the information supplied; `Output` names the result to report. You can follow the steps by hand.

```
Input: Returned Items with Damage Status (Yes/No)
Output: Damaged Count
Start
  For each Item in Returned Items
    Set Damaged Count to 0
    If Damage Status of Item is Yes
      Set Damaged Count to Damaged Count + 1
    End If
  End For
  Output Damaged Count
End
```

Which single change makes the algorithm correct for every permitted list? Leave all other instructions unchanged.

- A) Move `Set Damaged Count to 0` inside the `If`, immediately before adding `1`.
- B) Move `Set Damaged Count to 0` after `End For`, immediately before the output.
- C) Change `Damaged Count + 1` to `Damaged Count + 2`.
- D) Move `Set Damaged Count to 0` immediately after `Start`, before `For each`.

Question 7) Which pseudocode is equivalent to the flowchart?

`Stock Units` is the number of items currently available. It is a whole number that can be zero. The flowchart in Figure 1 produces one message. At the diamond, follow the arrow labeled with the answer to its Yes/No question. The symbol `<=` means “less than or equal to”. `Input` reads the supplied number, and `Output` reports the message.

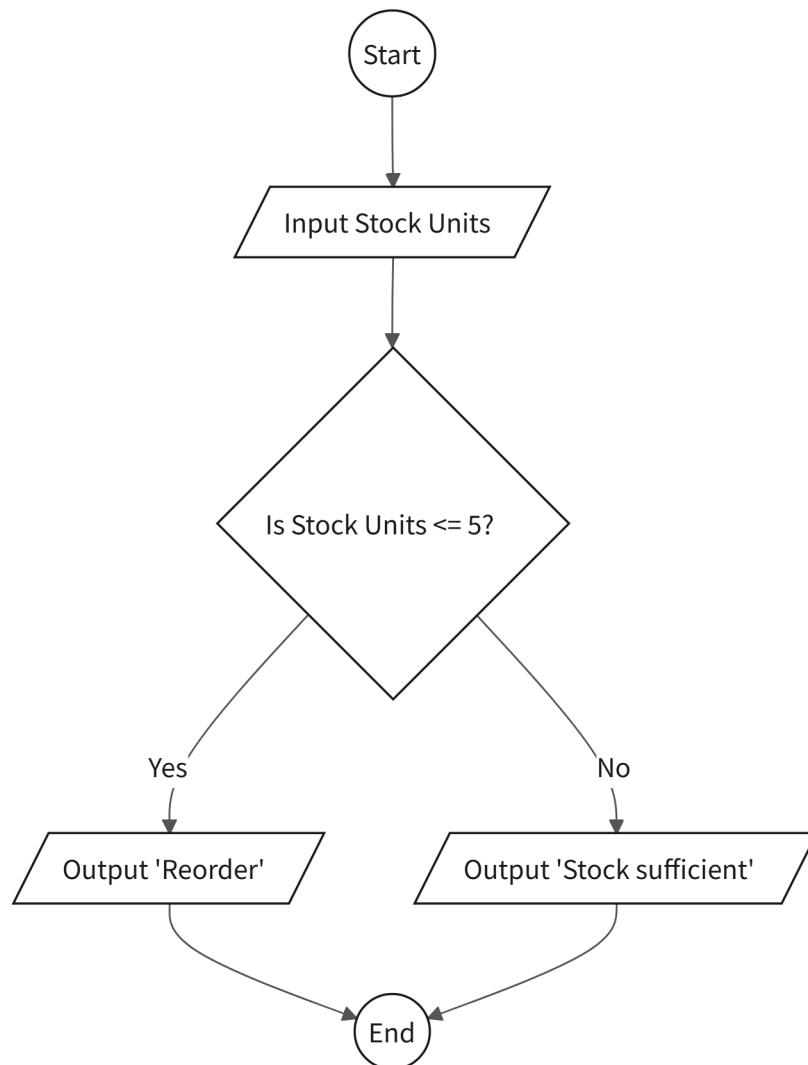


Figure 1: Flowchart for producing a replenishment message from the current stock.

Which pseudocode produces the same single message as the flowchart for **every permitted input**? An equivalent algorithm may phrase the decision differently.

A)

```
Input: Stock Units
Output: Replenishment Message
Start
  If Stock Units >= 5
    Output "Reorder"
  Else
    Output "Stock sufficient"
  End If
End
```

B)

```
Input: Stock Units
Output: Replenishment Message
Start
  If Stock Units > 5
    Output "Stock sufficient"
  Else
    Output "Reorder"
  End If
End
```

C)

```
Input: Stock Units
Output: Replenishment Message
Start
  If Stock Units <= 5
    Output "Stock sufficient"
  Else
    Output "Reorder"
  End If
End
```

D)

```
Input: Stock Units
Output: Replenishment Message
Start
  If Stock Units < 5
    Output "Reorder"
  Else
    Output "Stock sufficient"
  End If
End
```

Question 8) How does the running total change?

Trace the algorithm by following its steps for Values = [4, 2, 7] and recording how Total changes. For each visits 4, then 2, then 7. Set Total to Total + Value adds the current value to the total already stored.

```
Input: Values = [4, 2, 7]
Output: Total
Start
  Set Total to 0
  For each Value in Values
    If Value > 3
      Set Total to Total + Value
    End If
  End For
  Output Total
End
```

Which option gives Total **after each repetition**, in order, and the final output? Each arrow separates successive repetitions; it is not an additional calculation.

- A) Totals 4 -> 2 -> 7; final output 7.
- B) Totals 4 -> 0 -> 7; final output 7.
- C) Totals 4 -> 4 -> 11; final output 11.
- D) Totals 4 -> 6 -> 13; final output 13.

Question 9) What do these test results establish?

A teacher writes a set of instructions for labelling whole-number scores from 0 to 100, including both limits. The required rule, called the *specification*, is **Pass** for scores of 50 or higher and **Fail** for lower scores.

A tester follows the written instructions exactly for each score below. The expected result is what the required rule says; the actual result is what following the instructions produces. Assume there are no copying or calculation mistakes by the tester.

Table 3: Observed results of three independent classification tests.

Score being checked	Expected result	Actual result
20	Fail	Fail
60	Pass	Pass
75	Pass	Fail

You have the results but cannot see the teacher's instructions. Which conclusion is justified by these results alone?

- A) The instructions give the correct result for every score below 50 and the wrong result for every score above 50.
- B) The instructions give the wrong result for 75; the faulty step has not yet been identified.
- C) The expected result for 75 should change to **Fail** so that it agrees with the result obtained.
- D) The instructions must say "above 50" where they should say "50 or higher".

Question 10) Which test set checks the decision boundary?

A room holds at most 10 people. It contains 4 people before an arriving group asks to enter. The whole group must be admitted if everyone fits, including when the room becomes exactly full. Otherwise, the whole group must be rejected.

Group Size is a positive whole number. **Each test starts again with 4 people in the room**; groups from earlier tests are not still inside.

Which test set checks the largest group that can be admitted and its nearest smaller and larger group sizes, with all expected results correct?

Table 4: Candidate tests for the room-capacity rule; each arrow gives the expected result for one independent input.

Test set	Group sizes and expected results
A	9 -> Reject; 10 -> Reject; 11 -> Reject
B	5 -> Admit; 6 -> Admit; 7 -> Reject
C	1 -> Admit; 2 -> Admit; 3 -> Admit
D	5 -> Admit; 6 -> Reject; 7 -> Reject

- A) Test set A
- B) Test set B
- C) Test set C
- D) Test set D

Question 11) Which tests expose the grade-rounding algorithm's defect?

Use the University of Twente's published rounding schedule (17 June 2020) as the specification, meaning the required rules, for this exercise. It converts an unrounded course result to the final grade registered in Osiris. The schedule uses n to represent the whole-number part of the unrounded result. For example, when the result is 4.62, $n = 4$ and $n.25$ means 4.25. "At least" includes the first value, while "below" excludes the second value. Registered grades are written with one decimal place.

Table 5: Published University of Twente grade-rounding schedule used in this exercise.

Case	Unrounded course result	Grade registered in Osiris
n is not 5	At least $n.00$ and below $n.25$	$n.0$
n is not 5	At least $n.25$ and below $n.75$	$n.5$
n is not 5	At least $n.75$ and below $(n + 1).00$	$(n + 1).0$
$n = 5$	At least 5.00 and below 5.50	5.0
$n = 5$	At least 5.50 and below 6.00	6.0

The table is the specification. Use it to obtain the **expected** grade, then trace the proposed algorithm to obtain the **actual** grade.

```
Input: Unrounded Result
Output: Osiris Grade
Start
  Set n to the whole-number part of Unrounded Result
  Set Decimal Part to Unrounded Result - n
  If Decimal Part < 0.25
    Set Osiris Grade to n
  Else If Decimal Part < 0.75
    Set Osiris Grade to n + 0.5
  Else
    Set Osiris Grade to n + 1
  End If
  Output Osiris Grade
End
```

Each input -> expected output below describes one test. All listed expected outputs follow the schedule. A test fails when the actual and expected grades differ. Which option contains **two failing tests**?

- A) Test 5.24 -> 5.0 and test 5.25 -> 5.0.
- B) Test 5.00 -> 5.0 and test 5.99 -> 6.0.
- C) Test 5.49 -> 5.0 and test 5.50 -> 6.0.
- D) Test 5.74 -> 6.0 and test 5.75 -> 6.0.

Question 12) Plan a warehouse picking trip

Your goal: find a shortest walking trip that starts at D, collects one item from each of slots A, B, and C, and returns to D. This list of items is called *Order 17*.

What to hand in: two maps with distances, the completed trip-comparison table, and one shortest route with its total distance. Complete all three tasks. Drawings, calculations, and short sentences are sufficient.

The warehouse and movement rules

- Each grid cell is a square storage slot. The dark gray slots labeled A, B, and C contain the required items. The central rack has two back-to-back rows of slots, one accessible from each side.
- Use the measurements in the figure caption. D is halfway along the bottom boundary.
- From D, walk straight inward to the **middle of the front cross-aisle**. Then walk along the **middle of each aisle**. Turn where those middle lines meet. To collect an item, stop opposite the center of its slot. Travel is possible both ways; walking through racks or diagonally is not allowed.
- All items fit in one trip. Ignore collection time and sideways movement to reach a slot. You may pass a slot before collecting its item.

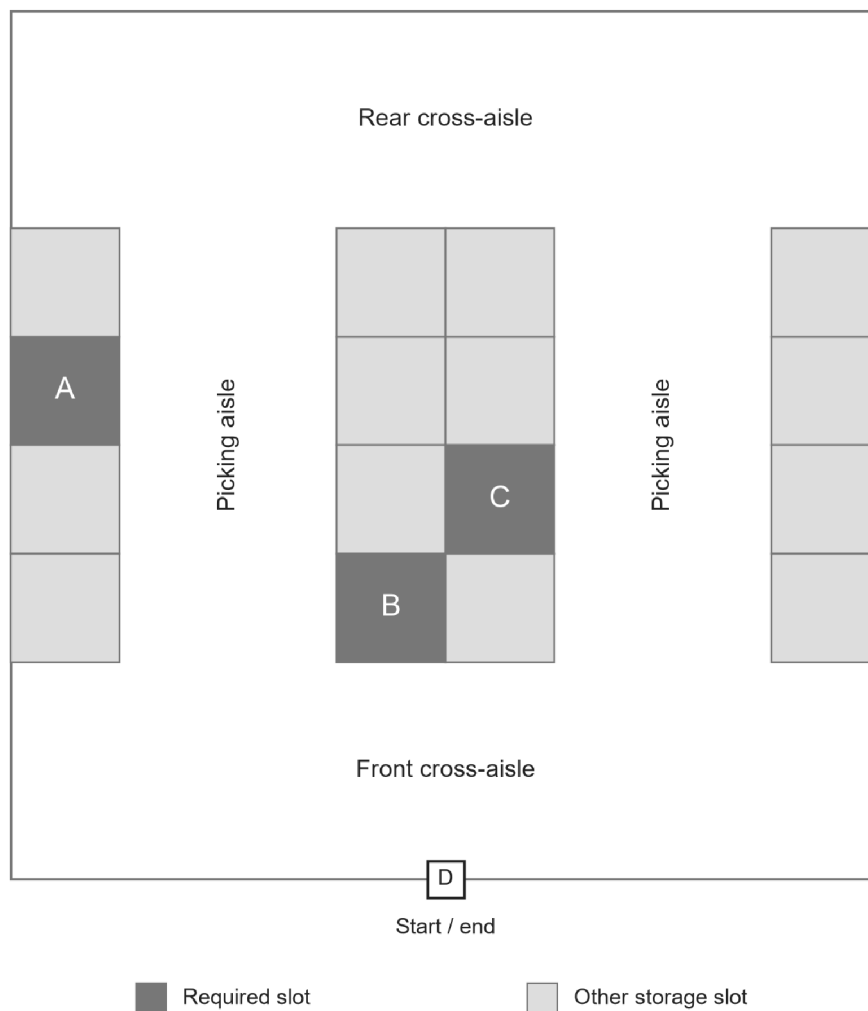


Figure 2: Warehouse layout for Order 17. Each square storage slot has a side length of 1 unit. Each picking aisle is 2 units wide, and the front and rear cross-aisles are also 2 units wide. Dark gray slots contain the required items A, B, and C; light gray cells are other storage slots. D marks the start and return point on the bottom boundary.

Warehouse case: required tasks

A *graph* is a map made of dots (*nodes*) and connecting lines (*edges*). The distance written on an edge is its *weight*. A *path* follows connected edges. Count each two-way connection once.

- 1. **Create the travel map.** Draw nodes at the collection stops for A, B, and C, at the aisle junctions, and at D. Include the connection from D to the front cross-aisle and its meeting point. Label the stops with their slot letters and give the junctions your own labels. Connect consecutive nodes along the aisle middles, with no other node between an edge's endpoints. Write every edge's distance in units and the numbers of nodes and edges. Explain why the collection nodes belong in the aisles rather than inside the rack cells.
- 2. **Simplify the map for Order 17.** Draw a second map keeping only D and the three collection stops. Join every pair and label each edge with the **shortest walking distance on your first map**. Write the numbers of nodes and edges. Show one actual aisle path for the connection between A and C and add its distances. Explain why a line on the second map does not permit walking straight through a rack.
- 3. **Find a shortest complete trip.** Complete the table below using your second map. For each row, add the distances of the four journeys, including departure from and return to D. Circle one shortest trip and trace its actual walking path on your first map. Report its total distance in units. Any equally short trip is acceptable.

Table 6: Compare all six collection orders, including the return to the depot.

Collection order	Complete trip	Total distance (units)
A, B, C	D, A, B, C, D	
A, C, B	D, A, C, B, D	
B, A, C	D, B, A, C, D	
B, C, A	D, B, C, A, D	
C, A, B	D, C, A, B, D	
C, B, A	D, C, B, A, D	

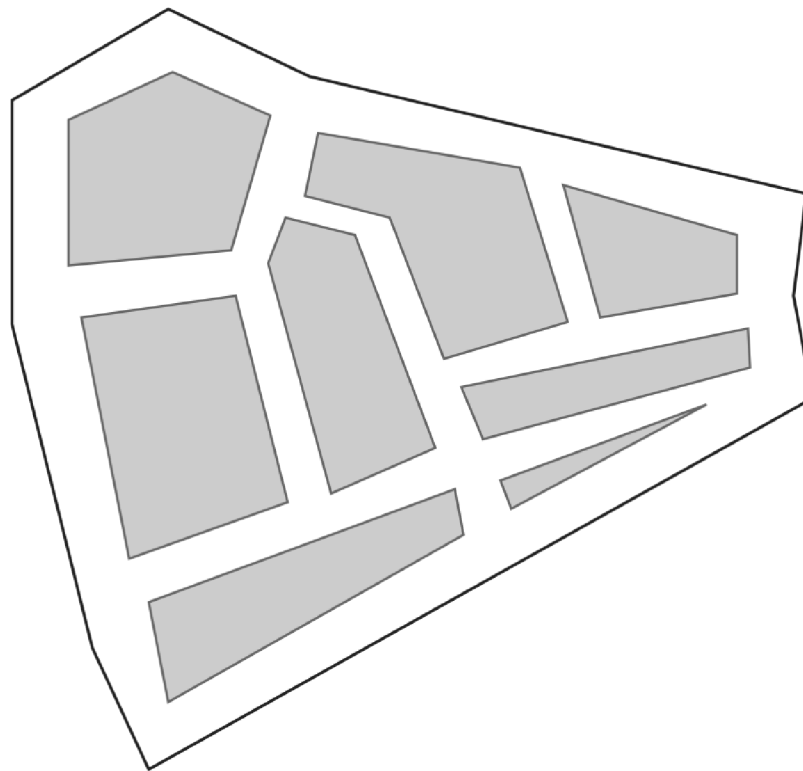
How a supplied Shortest Path procedure would help: given your first map and two nodes, it returns a shortest permitted path and its distance. You could use it once for each pair in Task 2 and keep those results for Task 3. For this small example, find the pairwise paths by hand; you do not need to describe how the supplied procedure works internally.

Question 13) Make a model of a street map

Your goal: turn the street layout below into a simpler map that could be used to find a shortest route between two places.

What to hand in: one labeled sketch and short answers to Tasks 1–3. You are designing the map, not calculating a shortest route: no measured street lengths or travel times are supplied.

The figure redraws a published street pattern. For this exercise, travel is allowed both ways along every street inside the outer outline. The gray blocks cannot be crossed. Do not add streets outside the outline.



Gray: blocks White between blocks: streets

Figure 3: Street pattern redrawn from Figure 1, lower-left panel, in Crucitti, Latora and Porta (2006). Gray areas are blocks; the white corridors between them are streets. The outline marks the extent of the street pattern. Drawing positions are not measured distances. *Source paper.*

A *graph* is a map made of dots (*nodes*) and connecting lines (*edges*). A *path* follows connected edges. A number attached to an edge, such as a distance, is called its *weight*.

1. **Build the model.** Draw a graph of the whole street layout. Explain what each node and edge represents. Give your nodes labels and count the nodes and edges; count a two-way connection once. Use nodes at street junctions, where a traveler has a choice of streets. An edge may bend without needing an extra node at the bend.
2. **Prepare it for shortest-distance routing.** Choose two different nodes as a start and a finish, and trace one possible path between them. What information must be added to the edges to decide which path is shortest? Explain how it would be used, and name one detail of the original picture that your model can omit.
3. **Add an arrival deadline.** The traveler now wants the shortest-distance route that arrives no later than a specified deadline. Assume street travel times are fixed during the trip. What extra information would you need, and how would you check whether a candidate route arrives on time? Explain why the shortest-distance route might no longer be acceptable.