

Abstract data types (HL only) · supported focus

Supported focus · HL

Supported scope

- Supported HL focus. First assessment 2027 target; official PDF returns 403; older acquired brief is final assessment 2026. Remaining guide, assessment and practical requirements retain their recorded holds.
- State model conditions and distinguish fictional data from measurements.

Adt · 1.1

Reasoning

A waiting-line simulation needs the oldest pending item first. A stack gives the most recent item first, so the operation contract matters. Start with a prediction. State the quantities or features you would compare, then decide what evidence could distinguish two explanations.

Adt · 2.1

Reasoning

An abstract data type specifies values and operations independently of a particular implementation. A stack is last in, first out; a queue is first in, first out. A collection interface needs defined behaviour for empty and invalid operations. stack: A last-in-first-out abstract data type; queue: A first-in-first-out abstract data type.

Adt · 3.1

Reasoning

Choose the structure for the required access pattern. Complexity depends on implementation and assumptions: removing the first item from a shifting array differs from advancing a head pointer in a queue. Trace operation sequences by hand, then compare a program with the expected states. Test empty, singleton and repeated operations. State overflow behaviour if capacity is fixed.

Adt · 4.1

Reasoning

State the known values and their units. Choose the relation because its assumptions fit this case, then rearrange before substitution. Known: enqueue A, B and C, then dequeue twice. A and B leave; C remains. After enqueue D there are two pending items, C then D. The same pushes on a stack would remove C then B.

Adt · 4.2

Reasoning

A queue starts empty, receives 7 items and removes 4. Find the number remaining.

Use the same sequence: known quantities $\rightarrow$ model $\rightarrow$ relation $\rightarrow$ substitution $\rightarrow$ unit and interpretation.

Adt · 5.1

Reasoning

An ADT is not a specific memory layout. A queue does not sort by priority unless its contract defines a priority queue. Return to the original observation. Explain what the result supports, which conditions it assumes, and one way to test a competing explanation.

Adt · 5.2

Reasoning

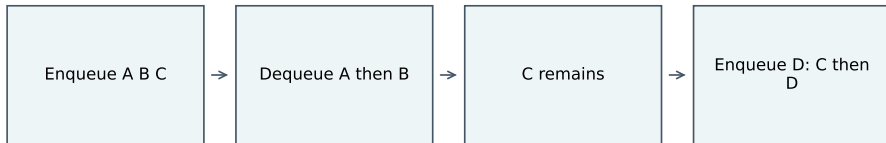
Every queue automatically sorts items by priority. This claim is false: An ADT is not a specific memory layout. A queue does not sort by priority unless its contract defines a priority queue.

Adt · 5.3

Reasoning

Abstract data types: an operation contract: Choose the structure for the required access pattern. Complexity depends on implementation and assumptions: removing the first item from a shifting array differs from advancing a head pointer in a queue.

Adt · diagram



- Original teaching diagram: Abstract data types: an operation contract
- Use the labels and the stated model together.

Adt · original transfer

Your turn

Start empty. Enqueue A,B; dequeue; enqueue C; dequeue. Give removals and final queue. Compare the same operations as pushes and pops on a stack.

Adt · transfer reasoning

Method and interpretation

Queue removes A then B; C remains. Stack removes B then C; A remains. An empty removal needs a defined exception or result, not invented data.

Adt · executable trace

- Trace the state changes and boundary guard.
- Run the boundary case and compare the printed output.

```
from collections import deque
queue = deque()
queue.extend(["A", "B"])
print(queue.popleft())
queue.append("C")
print(queue.popleft())
print(list(queue))
queue.clear()
print(queue.popleft() if queue else "EMPTY")
```

Check yourself

- Explain stack using the stated model.
- Explain queue using the stated model.
- Change one value in the original transfer; explain which assumption still matters.