

8.1 Abstract data types: an operation contract

Name: _____ Class: _____ Date: _____ Total: 10 marks

KEY WORDS `stack` • `queue`

Objective

Abstract data types: an operation contract. Supported HL concept preparation; original classroom practice, not an official IB paper or assessed learner work.

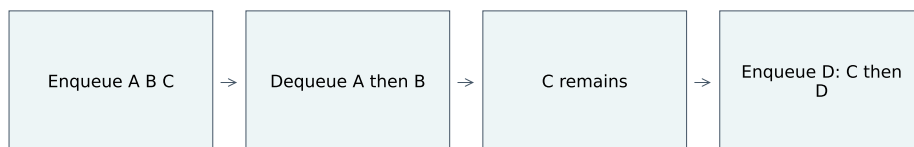
1 Key knowledge and worked examples

■ What would explain this observation?

- 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.

■ Build the model

- 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.



■ Choose evidence that can test it

- 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.

■ Work from known quantities

- 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.

Example:

A queue starts empty, receives 7 items and removes 4. Find the number remaining. Use the same sequence: known quantities → model → relation → substitution → unit and interpretation.

■ Check the conclusion and its limits

- 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.

Warn:

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.

Key:

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.

■ Runnable trace and boundary

```
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")
```

Expected output:

```
A
B
['C']
EMPTY
```

Removing from the left implements FIFO. The explicit empty guard defines the boundary result; deque.popleft without it would raise IndexError. A stack instead removes from the right.

2 Practice

2.1 Which structure models an ordinary arrival-order waiting line? Choose one answer.

- **A** Queue
- **B** Stack
- **C** An unordered choice with no contract

[1]

2.2 A queue starts empty, receives 7 items and removes 4. Find the number remaining. Show the relation, substitution and units.

[2]

2.3 Decide whether this claim is true and justify: Every queue automatically sorts items by priority.

[2]

2.4 A last-in-first-out abstract data type: write the technical term.

[1]

3 Exam-style questions

3.1 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.

[4]