1.4 DATA TYPES & STRUCTURES · 1.4.2

Lists & linked lists

List operations, nodes, pointers, traversal & insertion. Spec 1.4.2(b)(c).

01 List operations

Add	`append(x)` end · `insert(pos,x)`
Remove	`remove(x)` · `pop()/pop(pos)` returns it
Find	`index(x)` · `length()` · `isEmpty()`
Gotcha	Don't delete **while looping by index** → index out of range. Build a new list.

02 Linked list

Node	**data + pointer** to next node.
start	Holds position of first node.
End	Last node's pointer = **null**.
Access	Forwards only — must **traverse** (no index jump).

03 Operations

Traverse	`currentPtr = start; while ≠ null: print; move next`
Insert	prev → new node, new node → following node.
Delete	prev points **past** the removed node.
Key	Change **pointers**, not data.

04 Array vs linked list

Size	array fixed · linked list dynamic.
Memory	array contiguous · linked scattered + pointers.
Access	array direct (index) · linked sequential.
Insert	array shuffles · linked just repoints (cheap).

FINAL PASS BEFORE THE EXAM

Rapid exam tips

Six slips on linked-list questions.

01

A node = **data + pointer**. The last pointer is **null**.

02

Traverse from **start** following pointers — forwards only.

03

Insert: point the **new node at the next node first**, then redirect prev.

04

Delete: make prev point **past** the removed node.

05

Linked lists have **no direct index access** — only arrays do.

06

Don't delete from a list **while looping by index**.