Solutions

Each answer shows the steps, then the **result**.

2.1

- a **collection of data together with the operations** that can be performed on it (implementation hidden)

2.2

- stack = **LIFO** (Last In, First Out)
- queue = **FIFO** (First In, First Out)

2.3

- add = **push**
- remove = **pop**

2.4

- `Head` identifies the **first node** in the list
- the last node's pointer holds **NULL** (a null/sentinel pointer = end of list)

3.1

(i) test full, else increment `Top` and store

```
IF Top = 10 THEN
    OUTPUT "Stack full (overflow)"
ELSE
    Top ← Top + 1
    Stack[Top] ← NewItem
ENDIF
```

(ii) trying to **pop from an empty stack** (`Top = 0`) — there is nothing to remove

3.2

- popped: **D**, then **C** (LIFO order)
- now on top: **B**

3.3

(i) start at `Head`

- repeatedly output the current node's value and move to the node its `Next` pointer gives
- stop when the pointer is `NULL`

(ii) any one: items can be **inserted/deleted** without shifting others (just change pointers); the list can grow/shrink at run time