Solutions

Marking: **M1** a correct method, **A1** the correct answer, **B1** an independent mark.

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

2.1

7 first

then 9, then 1

left: `[4]`

2.2

A, then B

C is at the front

2.3

remove the first item: list $O(n)$, deque $O(1)$

read the middle item: list $O(1)$, deque $O(n)$

2.4

`new.next = p.next`

`p.next = new`

swapped, `p.next` already points to `new`, so `new.next = new` and the rest of the list is lost

3.1

(a)

```
def is_balanced(s):
    pairs = {")": "(", "]": "[", "}": "{"}
    stack = []
    for ch in s:
        if ch in "([{":
            stack.append(ch)
        elif ch in pairs:
            if not stack or stack.pop() != pairs[ch]:
                return False
    return not stack
```

push each opening bracket; on a closing bracket, check the stack is not empty; pop and compare with the matching opener; balanced only if the stack is empty at the end

(b) `{[()]}` is balanced; `([)]` fails at `)`, because the top of the stack is `[`

(c) $O(n)$: one push or pop per character

3.2

(a) `head = 2`; `tail`: after 4 it wraps, 5 goes to slot 4 and 6 to slot 0, so `tail = 1`; `count = 4`

(b) moving the elements would cost $O(n)$ per dequeue; the index arithmetic keeps both operations $O(1)$

(c) no: it is full when `count == 5`, and `count` is 4

3.3

(a) after the undos: `ab`, then `a`; after typing d: `ad`; redo does nothing, because typing d emptied the redo stack, so the text stays `ad`

(b) the old redo actions were made on a text that no longer exists, so redoing them would give nonsense

(c) $10^5 \times 1$ KB = **100 MB**; keep only the last 1000 actions, e.g. `deque(maxlen=1000)`, about 1 MB

(d) array lists (or a `deque`): push and pop at the end are $O(1)$ in both, but arrays use less memory and are faster in practice, and a `deque` with `maxlen` drops the oldest action for free

exercises.lol exercises.com.cn