Insertion Sort and Stack Applications

Lecture Script, Worked Traces, Worksheet and Solutions
Z5007 Programming and Data Structures, MTech DSAI | Week 3, Oct 19–22, 2026

1. Insertion Sort and Loop Invariants

> Insertion sort partitions the array into a sorted left prefix $a[0..i-1]$ and an unsorted right suffix $a[i..n-1]$. At each round, we pick $key = a[i]$, shift every element larger than key one position right, and drop key into the opened slot.

Its runtime equals $\Theta(n + I)$, where I is the number of inversions in the input. On nearly sorted data ($I = O(n)$), insertion sort finishes in linear time, which is why Python's Timsort switches to insertion sort on short subarrays.

2. The Stack ADT (LIFO): Arrays vs. Linked Nodes

A stack exposes four core $O(1)$ operations: `push(x)`, `pop()`, `peek()`, and `is_empty()`.

Table 1: Comparison of array-backed and linked-list-backed stack implementations.

Implementation	push / pop	Memory overhead	Cache locality
Dynamic Array Stack	$O(1)$ amortized	Contiguous buffer (spare slots)	High (contiguous reads)
Singly Linked Stack	$O(1)$ worst case	1 next pointer per item	Lower (pointer chasing)

```
def is_balanced(expr):
    pairs = {')': '(', ']': '[', '}': '{'}
    stack = []
    for ch in expr:
        if ch in '([{':
            stack.append(ch)
        elif ch in pairs:
            if not stack or stack.pop() != pairs[ch]:
                return False
    return len(stack) == 0
```

Check question. Trace `is_balanced("([)]")`. Where and why does it reject?

Answer. After pushing `'('` and `'['`, reading `')'` pops `'['`, which does not match pairs[`')'`]=`'('`, so it returns `False` at index 2.

3. Infix to Postfix and Expression Evaluation

Shunting-yard parses infix expressions using an operator stack comparing precedence, whereas a value stack evaluates any postfix token stream in one linear $O(n)$ pass without parentheses.

4. Worksheet (Students Complete Individually)

1. Trace insertion sort step by step on $[5, 2, 4, 6, 1, 3]$ and count total element shifts.
2. Convert infix expression $3 + 4 \times (2 - 1)$ into postfix using a stack, then evaluate the postfix stream.

5. Worksheet Solutions

1. Shifts per round ($i = 1..5$): $1 + 1 + 0 + 4 + 3 = 9$ shifts; final sorted list $[1, 2, 3, 4, 5, 6]$.
2. Postfix: `3 4 2 1 - * +`. Value evaluation: $2 - 1 = 1$, $4 \times 1 = 4$, $3 + 4 = 7$.

References

1. Canning, J., Broder, A. and Lafore, R. (2022). *Data Structures and Algorithms in Python*. Addison-Wesley. Chapters 3–4.
2. Cormen, T. H. et al. (2022). *Introduction to Algorithms*, 4th ed. MIT Press. Chapters 2 and 10.