19. Computational thinking and Problem-solving Starter Quiz · 课前小测

A-Level Computer Science Name: _____ Score: _____

1. A linear search:
   - ☐ checks each element in turn and works on any list
   - ☐ requires the data to be sorted
   - ☐ always finds the target in one step
   - ☐ only works on numbers
2. Bubble sort works by:
   - ☐ repeatedly swapping adjacent out-of-order pairs
   - ☐ inserting each element into a sorted prefix
   - ☐ halving the list each time
   - ☐ building a binary tree
3. Which Big-O describes binary search?
   - ☐ O(log n)
   - ☐ O(n)
   - ☐ $O(n^2)$
   - ☐ O(n log n)
4. Every recursive algorithm must have a base case because:
   - ☐ without it the recursion never stops
   - ☐ it makes the code shorter
   - ☐ it speeds up the program
   - ☐ the compiler requires a comment
5. The worst-case time complexity of binary search is:
   - ☐ O(log n)
   - ☐ O(n)
   - ☐ $O(n^2)$
   - ☐ O(1)
6. What does Factorial(4) return?

   _____

7. About how many comparisons does a binary search need for one million sorted items?

   _____

8. Insertion sort runs close to O(n) on small or nearly-sorted arrays, because few elements need to be shifted.

   ☐ True ☐ False

9. A stack (LIFO) naturally drives a depth-first traversal, while a queue (FIFO) drives a breadth-first traversal.

   ☐ True ☐ False

10. Recursion is a natural fit for self-similar problems (trees, divide-and-conquer), but each call adds a stack frame — so without a base case it overflows the stack.

    ☐ True ☐ False

A-Level Computer Science

1. **checks each element in turn and works on any list**

   Linear search needs no preparation and works on any list, at worst O(n).

2. **repeatedly swapping adjacent out-of-order pairs**

   Each pass swaps adjacent pairs, ”bubbling” the largest element to the end.

3. **O(log n)**

   Halving the range each step is logarithmic — O(log n).

4. **without it the recursion never stops**

   The base case is the condition that ends the chain of calls; without it the recursion runs forever.

5. **O(log n)**

   Halving the range each step gives a logarithmic number of comparisons.

6. **24**

   $4 \times 3 \times 2 \times 1 = 24$.

7. **20**

   $\log_2(1\,000\,000) \approx 20$ — about 20 comparisons.

8. **True**

   On nearly-sorted data each new item is already almost in place — which is why insertion sort beats fancier sorts on small inputs.

9. **True**

   The ADT you choose decides the search order — stack goes deep first, queue explores level by level.

10. **True**

    A simple counting loop is cleaner for plain iteration; recursion shines when the problem contains smaller copies of itself.