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. Linear search is the better choice when the data is:
   - ☐ unsorted or only a small list
   - ☐ very large and sorted
   - ☐ in a database index
   - ☐ already a binary tree
6. 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

7. An O(n squared) algorithm takes 4 seconds on 1,000 items. Roughly how many seconds will it take on 2,000?

   _____
8. What must a recursive routine have to terminate? Select **all** that apply.
   - ☐ a base case that is solved directly without another call
   - ☐ an input that gets smaller on each recursive call
   - ☐ a path that actually reaches the base case
   - ☐ a loop inside the recursive case

   Tick every correct option.
9. About how many comparisons does a binary search need for one million sorted items?

   _____
10. What does Factorial(4) return?

    _____

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. **unsorted or only a small list**

   With no order to exploit (or a tiny list), linear search avoids the cost of sorting first.

6. **True**

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

7. **16**

   Doubling n quadruples an O(n squared) time. The same doubling would take an O(n) algorithm from 4 seconds to 8.

8. **a base case that is solved directly without another call; an input that gets smaller on each recursive call; a path that actually reaches the base case**

   A base case alone is not enough: if the input never shrinks, the base case is never reached and frames pile up until the stack overflows.

9. **20**

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

10. **24**

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