3 Iteration, lists, and efficiency – Part 2

Name: _____ Class: _____ Date: _____

Recall

In Part A you met variables and choices. Now repeating steps and handling many values:

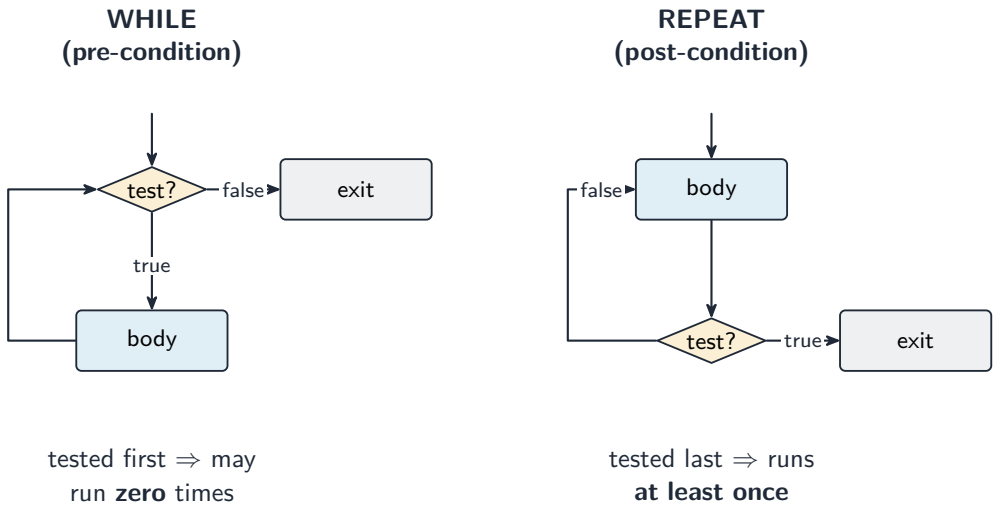
- Doing something 100 times by hand is slow; a loop does it fast.
- A list holds many values under one name.
- Some methods are much faster than others.

Each idea has a worked example before the Practice.

New ideas

1 Iteration

Iteration (a loop) repeats steps. `REPEAT 5 TIMES` runs its block five times; a conditional loop repeats while a test stays true.



tested first ⇒ may run **zero** times

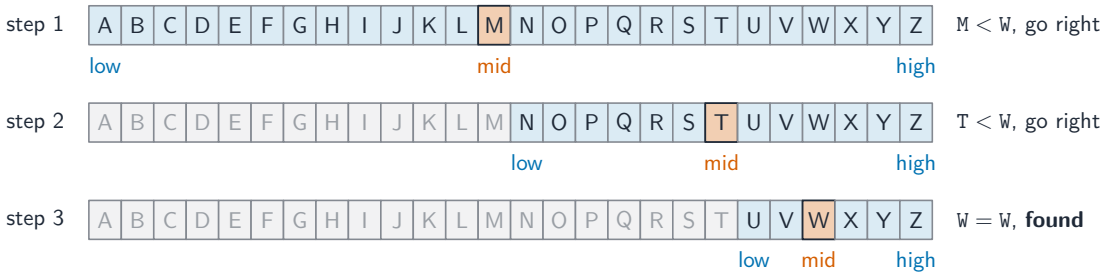
tested last ⇒ runs **at least once**

2 Lists

A **list** stores many values in order under one name, each reached by its position (index).

■ 3 Binary search

To find a value in a **sorted** list, **binary search** checks the middle and throws away the half that cannot contain the target, repeating.



Worked example. A sorted list of 8 items needs at most 3 checks ($8 \to 4 \to 2 \to 1$), far fewer than checking all 8.

■ 4 Efficiency

An algorithm's **efficiency** is how its work grows with the input size. A **reasonable-time** method grows slowly enough to be practical; an **unreasonable-time** one grows so fast it becomes impossible for large inputs.

Watch out: binary search is only fast because it **halves** the list each step. It only works on a **sorted** list – on an unsorted one you must use the slower linear search.

Practice

Try these in order. The methods are all above.

2.1 State what iteration (a loop) does. [1]

2.2 State what a list is. [2]

2.3 A sorted list has 16 items. State roughly how many checks binary search needs. [2]

2.4 State the one requirement binary search has of the list. [1]

2.5 Explain the difference between a reasonable-time and an unreasonable-time algorithm. [2]