9 Algorithm Design and Problem-solving – Part 1

Name: _____ Class: _____ Date: _____

Vocabulary 词汇

Write each word three times. 把每个单词抄写三遍。

English	中文	Write it 3 times (English)		
Computational thinking	计算思维			
decomposition	分解			
abstraction	抽象			
algorithm	算法			
sequence	顺序			
selection	选择			
iteration	迭代			
flowchart	流程图			
pseudocode	伪代码			
variable	变量			

Recall

You have solved problems step by step before. Bring this with you:

- A clear set of steps can be followed by someone else to get the same result.
- You have read simple flowcharts.

At AS we learn to think like a computer scientist and to write algorithms carefully —the planning that comes before any code.

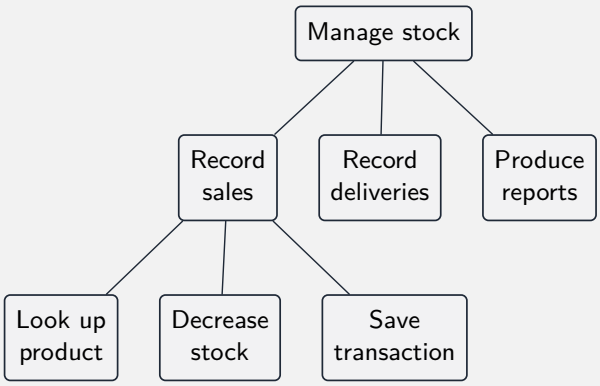
New ideas

Read these first. Each idea has a short worked example. Then do the Practice.

1 Computational thinking

Computational thinking 计算思维 means solving a problem in a clear, step-by-step way that a computer could follow. Two key skills:

- **decomposition** 分解: break a big problem into smaller, easier parts;
- **abstraction** 抽象: keep only the details that matter and ignore the rest.

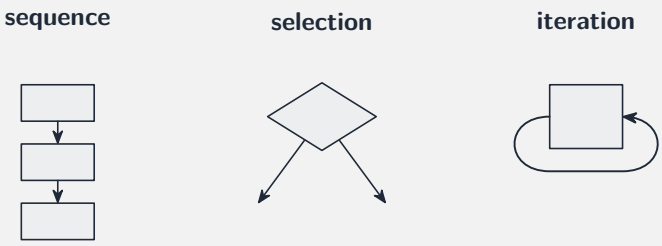


Worked example. To build a quiz app, decomposition splits it into "ask a question", "check the answer" and "keep the score"; abstraction ignores the screen colour and font, which do not affect how it works.

2 Algorithms and the three building blocks

An **algorithm** 算法 is a set of clear, ordered steps that always finishes. Every algorithm is built from just three structures:

- **sequence** 顺序: steps done in order;
- **selection** 选择: choose between paths with `IF`;
- **iteration** 迭代: repeat steps with a loop.

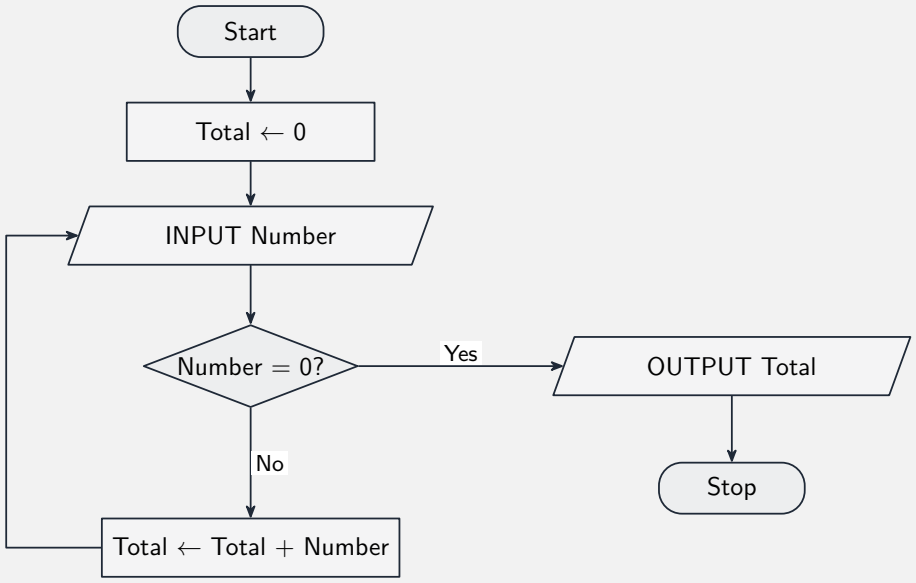


Worked example. "Keep asking for a password until it is correct" uses all three: read the input (sequence), check it (selection), and repeat while it is wrong (iteration).

Watch out: an algorithm must always *finish*. A loop needs a condition that eventually becomes false —or it runs forever.

■ 3 Writing it down

We plan an algorithm as a **flowchart** 流程图 or in **pseudocode** 伪代码 (code-like English).



Pseudocode stores values in a **variable** 变量:

```
INPUT age
IF age >= 18 THEN
    OUTPUT "adult"
ELSE
    OUTPUT "minor"
ENDIF
```

Worked example. If `age` is 20, the test `age >= 18` is true, so this outputs ”adult”.

Practice

9.1 Explain what decomposition means. [1]

9.2 Explain what abstraction means. [1]

9.3 State which building block an `IF ⋯ELSE` statement is. [1]

9.4 Name the three building blocks of any algorithm. [3]

9.5 Write pseudocode that inputs two numbers and outputs their sum. [3]

9.6 Explain why breaking a large program into smaller parts (decomposition) makes it easier to write and test. [2]

9.7 Challenge. Write pseudocode that keeps inputting numbers until a 0 is entered, then outputs their total. [3]