AP COMPUTER SCIENCE PRINCIPLES • EXERCISE SHEET

3.12 Calling Procedures

Name: _____ Class: _____ Date: _____ **Total: 8 marks**

Objective

Build the skills to answer exam questions on **calling procedures**.

You must be able to:

- explain how a **procedure** 过程 (function or method) is a named block of reusable code
- **call** a procedure by name to run its instructions
- pass **arguments** 实参 through its parameters
- use a **return value** 返回值 in a later expression

1 Worked examples

Study these first. Each one shows the method for a question type used later.

Procedures

A **procedure** (also called a function or method) is a **named block of reusable code**. You **call** it by name to run its instructions from anywhere.

Arguments and return values

Values passed in are **arguments**, received through the procedure's **parameters**. A procedure may produce a **return value** that the caller uses.

Abstraction

Procedures support **abstraction** by hiding their internal detail from the caller: `area(5, 3)` returns 15 without the caller seeing how.

2 Practice

2.1 Define a procedure. [1]

2.2 State how you run a procedure from elsewhere in a program. [1]

2.3 State what a return value is. [1]

---

3 Exam-style questions

3.1 A procedure is [1]

- **A** a single variable
- **B** a named block of reusable code
- **C** a loop
- **D** a data type

3.2 Values passed into a procedure are called [1]

- **A** return values
- **B** arguments
- **C** comments
- **D** lists

3.3 A procedure `double(n)` returns `n` × `2`.

(a) State the value of `double(6)`. [1]

(b) Name what `6` is when it is passed in. [1]

(c) State how procedures support abstraction. [1]

4 Go further

- work through the **3.12 Calling Procedures** lesson on the **Learn** page;
- read the **Algorithms and Programming** section of the AP Computer Science Principles handout on the **Know** page.

Solutions

2.1 a named block of reusable code (a function or method).

2.2 call it by its name.

2.3 a value that a procedure produces and hands back to the caller.

3.1 B.

3.2 B.

3.3 (a) 12. (b) an argument. (c) they hide the implementation detail from the caller.