

11.1.2 Built-in functions and library routines

Name: _____ Class: _____ Date: _____ **Total: 48 marks**

KEY WORDS built-in functions 内置函数 • library routines 库例程 • parameters 参数
• concatenates 拼接 • program library 程序库

Objective

Learn to use the **built-in functions** 内置函数 and **library routines** 库例程 of pseudocode: the string functions, the conversion functions and the numeric functions that the Paper 2 **insert** 附页 gives you. This is the last sheet of 11.1. You need the variables of sheet 11.1 and the operators of sheet 11.1.1.

You must be able to:

- use the string functions `LENGTH`, `LEFT`, `RIGHT`, `MID`, `TO_UPPER` and `TO_LOWER`
- join strings with `&`, and convert between numbers and strings with `NUM_TO_STR` and `STR_TO_NUM`
- use `ASC`, `CHR`, `INT` and `RAND`
- work out an expression in which one function is inside another
- state the benefits of using a program library

1 Worked examples

Study these first. Each one shows a **method** that a question later on this sheet needs.

■ What a function is

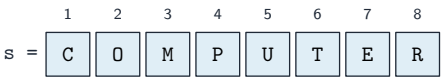
A **function** takes values (its **parameters** 参数), does a job, and **returns** a result. The result **replaces** the call:

`Size ← LENGTH("Computer")` stores 8 in `Size`.

The exam gives this list on the insert: do not learn it by heart, but use the **exact names** and the **correct order** of the parameters.

■ The string functions

- Positions are counted from **1**, and a space counts as a character: `LENGTH("Happy Days")` is 10.
- `TO_UPPER` and `TO_LOWER` take a **CHAR** or a **STRING**. The Pseudocode Guide also has `UCASE` and `LCASE`, which take **one CHAR only**.



`LENGTH(s)` = 8
`LEFT(s, 3)` = "COM"
`MID(s, 4, 3)` = "PUT"

`RIGHT(s, 2)` = "ER"
`TO_UPPER(s)` = "COMPUTER"
`TO_LOWER(s)` = "computer"

function	returns	example
<code>LENGTH(s)</code>	the number of characters in <code>s</code>	<code>LENGTH("Computer")</code> is 8
<code>LEFT(s, n)</code>	the first <code>n</code> characters	<code>LEFT("Computer", 3)</code> is "Com"
<code>RIGHT(s, n)</code>	the last <code>n</code> characters	<code>RIGHT("Computer", 2)</code> is "er"
<code>MID(s, start, n)</code>	<code>n</code> characters, beginning at position <code>start</code>	<code>MID("Computer", 4, 3)</code> is "put"
<code>TO_UPPER(x)</code>	<code>x</code> in capital letters	<code>TO_UPPER("hi")</code> is "HI"
<code>TO_LOWER(x)</code>	<code>x</code> in small letters	<code>TO_LOWER("Hi")</code> is "hi"

■ Joining and converting

- The operator `&` **concatenates** 拼接 (joins) two strings: `"Sum" & "mer"` is "Summer".
- `+` does **not** join strings, and a number cannot be joined to a string. Convert it first:

function	returns	example
<code>NUM_TO_STR(x)</code>	the number <code>x</code> as a string	<code>NUM_TO_STR(87.5)</code> is "87.5"
<code>STR_TO_NUM(s)</code>	the string <code>s</code> as a number	<code>STR_TO_NUM("23.45")</code> is 23.45
<code>IS_NUM(s)</code>	TRUE if <code>s</code> is a valid number	<code>IS_NUM("12a")</code> is FALSE
<code>ASC(c)</code>	the ASCII code of the character <code>c</code>	<code>ASC('A')</code> is 65
<code>CHR(n)</code>	the character with the ASCII code <code>n</code>	<code>CHR(66)</code> is 'B'

"Total: " & `Score` is an error when `Score` is an **INTEGER**. The correct statement is "Total: " & `NUM_TO_STR(Score)`.

■ Numeric functions

function	returns	example
INT(x)	the whole-number part of x	INT(27.5415) is 27
RAND(n)	a random real number from 0 up to, but not including, n	RAND(6) could be 4.27

RAND returns a real number, so a random **integer** needs INT. Dice, 1 to 6: INT(RAND(6)) gives 0 to 5, so add 1. From Low to High: INT(RAND(High - Low + 1)) + Low.

■ A function inside a function

Work from the **inside out**: work out the inner function first, and put its result in the outer one.

Worked example: TO_UPPER(RIGHT("Program", 2)).

1. Inner: RIGHT("Program", 2) is "am".
2. Outer: TO_UPPER("am") is "AM".

Worked example: LENGTH("Hello") + STR_TO_NUM("4"). LENGTH("Hello") is 5 and STR_TO_NUM("4") is 4, so the result is 9.

■ Program libraries

A **program library** 程序库 is a collection of routines that are already written, tested and compiled. Benefits of using one:

- it **saves time**: no new code to write and test
- the routines are already **tested**, so they are reliable
- they are written by **experts**, and may do a job the programmer could not write
- a routine can be **used many times**

2 Practice

Now use the methods above. The first questions need one idea each.

2.1 Name the function that does each job. [5]

- (a) returns the number of characters in a string
- (b) returns the last few characters of a string
- (c) changes a string into a number
- (d) returns the ASCII code of a character
- (e) returns the whole-number part of a real number

2.2 Give the value of each expression. [4]

- (a) LENGTH("Computer")
- (b) RIGHT("Computer", 3)
- (c) MID("Computer", 4, 2)
- (d) INT(STR_TO_NUM("9.7"))

3.5 State **three** benefits of using a program library rather than writing the routines yourself. [3]

3.6 A shop gives every product a code such as "TV-2025-0457". The code has three parts: two letters for the type of product, the year the product was first sold, and a product number. The code is stored in the variable `Code`.

(a) Work out the value of each expression when `Code` is "TV-2025-0457". [3]

Expression	Evaluates to
LEFT(<code>Code</code> , 2)	
STR_TO_NUM(MID(<code>Code</code> , 4, 4)) + 1	
LENGTH(<code>Code</code>) - 8	

(b) Write **one** pseudocode statement that stores the product number, as an `INTEGER`, in the variable `ProductNumber`. [3]

(c) The shop makes a short label from the code: the two letters in small letters, then the last two digits of the year. For "TV-2025-0457" the label is "tv25". Write **one** pseudocode statement that stores the label in the variable `Label`. [4]
