

11.1.1 Writing pseudocode from a design, and the library routines

Name: _____ Class: _____ Date: _____

Total: 32 marks

KEY WORDS pseudocode 伪代码 • variables 变量 • flowchart 流程图 • constant 常量
• program library 程序库

Objective

Build the skill that sheet 11.1 never asks for. That sheet drills declarations, operators and expressions in isolation; the **first** statement of 11.1 is to write **pseudocode from a given design** presented as a **program flowchart** 流程图 or as **structured English** 结构化英语, and the words “flowchart” and “structured English” do not appear in it at all. This sheet is that conversion, plus the **library routines** 库例程 the Paper 2 insert supplies and the parent sheet mostly leaves out.

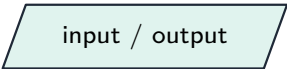
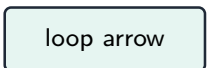
You must be able to:

- write pseudocode from a design given as a program flowchart or as structured English
- declare and initialise **constants** 常量 and variables, and write complete, closed constructs
- use the built-in functions and library routines with their exact names and parameter order
- 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.

- Every flowchart symbol has one pseudocode form

flowchart symbol	pseudocode
 input / output	INPUT / OUTPUT
 decision	IF...THEN or CASE
 process	x = expression
 loop arrow	WHILE / FOR / REPEAT

In the design	In pseudocode
a parallelogram	INPUT x or OUTPUT x
a rectangle	an assignment, x <- expression
IF ... THEN	

... ELSE ...

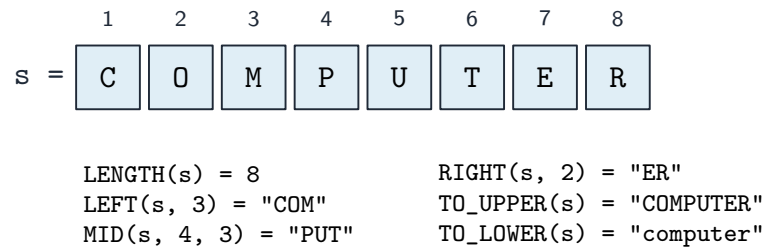
ENDIF | a diamond with several exits on one value | CASE OF ... OTHERWISE ...
 ENDCASE | | an arrow looping **back above** the body | REPEAT ... UNTIL – the test
 is after the body | | a diamond **above** the body it controls | WHILE ... ENDWHILE –
 the test is before the body | | a counter set, incremented and tested | FOR ... NEXT
 |

■ What “complete” means, and where the marks are

A conversion question is marked on things that are easy to leave out:

1. **Declare every variable**, with its type: DECLARE Total : INTEGER. A constant is CONSTANT Pi = 3.14159 and is declared once, never assigned to again.
2. **Initialise every total and counter** before the loop. Total <- 0 is a mark on its own.
3. **Close every construct**: ENDIF, ENDWHILE, NEXT, ENDCASE. An unclosed construct costs the structure mark.
4. **Use the design’s own identifiers**. Renaming the variables loses the link to the design the question gave you.

■ The library routines, and the trap in their names



Routine	Returns
LENGTH(s)	the number of characters in s
LEFT(s, n) / RIGHT(s, n)	the first / last n characters
MID(s, start, n)	n characters from position start – positions count from 1
TO_UPPER(x) / TO_LOWER(x)	x in capitals / in small letters; x may be a CHAR or a STRING
UCASE(c) / LCASE(c)	one CHAR in upper / lower case – Pseudocode Guide only, not on the insert
NUM_TO_STR(x) / STR_TO_NUM(s)	a number as a string / a string as a number
IS_NUM(s)	TRUE if s is a valid number
ASC(c) / CHR(n)	the character code of c / the character with code n
INT(x)	the whole-number part of x
RAND(n)	a random real from 0 up to, but not including, n

- **The two case routines come from two different documents.** TO_UPPER and TO_LOWER are on the **Paper 2 insert** and take a **CHAR** or a **STRING**. UCASE and LCASE are in the **Pseudocode Guide** and take one **CHAR** only; the insert does not list them. So on Paper 2, use TO_UPPER, and never apply UCASE to a whole word.
- **&** joins strings. **+** does **not**: "Done" + LENGTH(s) is invalid, and the correct form is "Done" & NUM_TO_STR(LENGTH(s)).
- RAND returns a real, so a random **integer** needs INT. From 1 to 6 is INT(RAND(6)) + 1; from Low to High inclusive is INT(RAND(High - Low + 1)) + Low.
- In a nested call the **inner** function runs first: TO_UPPER(RIGHT("Program", 2)) is TO_UPPER("am"), which is "AM".

■ A worked conversion, from structured English

Write pseudocode for: input 10 marks one at a time, keep a running total, and output the average.

```
DECLARE Count, Mark, Total : INTEGER
DECLARE Average : REAL
Total <- 0
FOR Count <- 1 TO 10
    OUTPUT "Enter a mark:"
    INPUT Mark
    Total <- Total + Mark
NEXT Count
Average <- Total / 10
OUTPUT "Average is ", Average
```

Five things earn marks and each is visible on its own line: the **declarations**, **Total <- 0** **before** the loop, the loop running the right number of times, the accumulation inside it, and the average calculated and output **after** NEXT.

2 Practice

Now use the methods above.

2.1 Give the pseudocode construct that each flowchart feature becomes.

[4]

- (a) a parallelogram containing **Name**
- (b) a rectangle containing **Total = Total + 1**
- (c) a diamond with **Yes** and **No** exits that rejoin below
- (d) a diamond **below** the body, whose **Yes** exit loops back up

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"))`

2.3 Write the pseudocode statement that assigns to `Dice` a random integer from 1 to 6, and state why `INT` is needed.

[2]

2.4 A student writes `Message <- "Total: " + Score`, where `Score` is an `INTEGER`. State what is wrong and write the correct statement.

[2]

3 Exam-style questions

3.1 An algorithm is described in structured English:

1. Set a running total to zero.
2. Input a temperature. Repeat this until the value -99 is entered.
3. Add each temperature except -99 to the running total, and count how many were entered.
4. Output the total and the number of temperatures.

Write pseudocode for this algorithm. Declare your variables.

[5]

3.2 A program flowchart has this structure: **Start**; a process box setting **Count** to 1; a parallelogram **INPUT Name**; a process box **OUTPUT Name**; a process box adding 1 to **Count**; a diamond **below** those boxes reading **Count** ≤ 5 ? whose **Yes** exit loops back to the **INPUT**; and on the **No** exit, **Stop**.

Write the pseudocode for this flowchart.

[5]

3.3 Give the value of each expression.

[3]

(a) `TO_UPPER(RIGHT("Program", 2))`

(b) `LENGTH(MID("Computer", 3, 4))`

(c) `ASC('Q') - ASC('A')`

3.4 A school builds each student's username from their surname, their first name and their year group: the **first three letters of the surname in capitals**, then the **first letter of the first name**, then the **year group**.

Write a single pseudocode statement that assigns this username to `Username`, using the variables `Surname`, `FirstName` and `Year`, where `Year` is an `INTEGER`.

[4]

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

[3]
