

Appendix 5: Pseudocode command set

Questions in the written examination that involve code will use this pseudocode for clarity and consistency. However, students may answer questions using any valid method.

Data types

INTEGER

REAL

BOOLEAN

CHARACTER

Type coercion

Type coercion is automatic if indicated by context. For example $3 + 8.25 = 11.25$
(integer + real = real)

Mixed mode arithmetic is coerced like this:

	INTEGER	REAL
INTEGER	INTEGER	REAL
REAL	REAL	REAL

Coercion can be made explicit. For example, RECEIVE age FROM (INTEGER) KEYBOARD assumes that the input from the keyboard is interpreted as an INTEGER, not a STRING.

Constants

The value of constants can only ever be set once. They are identified by the keyword CONST. Two examples of using a constant are shown.

```
CONST REAL PI
```

```
SET PI TO 3.14159
```

```
SET circumference TO radius * PI * 2
```

Data structures

ARRAY

STRING

Indices start at zero (0) for all data structures.

All data structures have an append operator, indicated by &.

Using & with a STRING and a non-STRING will coerce to STRING. For example, SEND 'Fred' & age TO DISPLAY, will display a single STRING of 'Fred18'.

Identifiers

Identifiers are sequences of letters, digits and `'_'`, starting with a letter, for example: `MyValue`, `myValue`, `My_Value`, `Counter2`

Functions

`LENGTH()`

For data structures consisting of an array or string.

`RANDOM(n)`

This generates a random number from 0 to n.

Comments

Comments are indicated by the `#` symbol, followed by any text.

A comment can be on a line by itself or at the end of a line.

Devices

Use of `KEYBOARD` and `DISPLAY` are suitable for input and output.

Additional devices may be required, but their function will be obvious from the context. For example, `CARD_READER` and `MOTOR` are two such devices.

Notes

In the following pseudocode, the `< >` indicates where expressions or values need to be supplied. The `< >` symbols are not part of the pseudocode.

Variables and arrays		
Syntax	Explanation of syntax	Example
SET Variable TO <value>	Assigns a value to a variable.	SET Counter TO 0 SET MyString TO 'Hello world'
SET Variable TO <expression>	Computes the value of an expression and assigns to a variable.	SET Sum TO Score + 10 SET Size to LENGTH(Word)
SET Array[index] TO <value>	Assigns a value to an element of a one-dimensional array.	SET ArrayClass[1] TO 'Ann' SET ArrayMarks[3] TO 56
SET Array TO [<value>, ...]	Initialises a one-dimensional array with a set of values.	SET ArrayValues TO [1, 2, 3, 4, 5]
SET Array [RowIndex, ColumnIndex] TO <value>	Assigns a value to an element of a two-dimensional array.	SET ArrayClassMarks[2,4] TO 92

Selection		
Syntax	Explanation of syntax	Example
IF <expression> THEN <command> END IF	If <expression> is true then command is executed.	IF Answer = 10 THEN SET Score TO Score + 1 END IF
IF <expression> THEN <command> ELSE <command> END IF	If <expression> is true then first <command> is executed, otherwise second <command> is executed.	IF Answer = 'correct' THEN SEND 'Well done' TO DISPLAY ELSE SEND 'Try again' TO DISPLAY END IF

Repetition		
Syntax	Explanation of syntax	Example
<pre>WHILE <condition> DO <command> END WHILE</pre>	<p>Pre-conditioned loop. Executes <command> whilst <condition> is true.</p>	<pre>WHILE Flag = 0 DO SEND 'All well' TO DISPLAY END WHILE</pre>
<pre>REPEAT <command> UNTIL <expression></pre>	<p>Post-conditioned loop. Executes <command> until <condition> is true. The loop must execute at least once.</p>	<pre>REPEAT SET Go TO Go + 1 UNTIL Go = 10</pre>
<pre>REPEAT <expression> TIMES <command> END REPEAT</pre>	<p>Count controlled loop. The number of times <command> is executed is determined by the expression.</p>	<pre>REPEAT 100-Number TIMES SEND '*' TO DISPLAY END REPEAT</pre>
<pre>FOR <id> FROM <expression> TO <expression> DO <command> END FOR</pre>	<p>Count controlled loop. Executes <command> a fixed number of times.</p>	<pre>FOR Index FROM 1 TO 10 DO SEND ArrayNumbers[Index] TO DISPLAY END FOR</pre>
<pre>FOR <id> FROM <expression> TO <expression> STEP <expression> DO <command> END FOR</pre>	<p>Count controlled loop using a step.</p>	<pre>FOR Index FROM 1 TO 500 STEP 25 DO SEND Index TO DISPLAY END FOR</pre>
<pre>FOR EACH <id> FROM <expression> DO <command> END FOREACH</pre>	<p>Count controlled loop. Executes for each element of an array.</p>	<pre>SET WordsArray TO ['The', 'Sky', 'is', 'grey'] SET Sentence to "" FOR EACH Word FROM WordsUArray DO SET Sentence TO Sentence & Word & '' END FOREACH</pre>

Input/output		
Syntax	Explanation of syntax	Example
SEND <expression> TO DISPLAY	Sends output to the screen.	SEND 'Have a good day.' TO DISPLAY
RECEIVE <identifier> FROM (type) <device>	Reads input of specified type.	RECEIVE Name FROM (STRING) KEYBOARD RECEIVE LengthOfJourney FROM (INTEGER) CARD_READER RECEIVE YesNo FROM (CHARACTER) CARD_READER

File handling		
Syntax	Explanation of syntax	Example
READ <File> <record>	Reads in a record from a <file> and assigns to a <variable>. Each READ statement reads a record from the file.	READ MyFile.doc Record
WRITE <File> <record>	Writes a record to a file. Each WRITE statement writes a record to the file.	WRITE MyFile.doc Answer1, Answer2, 'xyz 01'

Subprograms		
Syntax	Explanation of syntax	Example
PROCEDURE <id> (<parameter>, ...) BEGIN PROCEDURE <command> END PROCEDURE	Defines a procedure.	PROCEDURE CalculateAverage (Mark1, Mark2, Mark3) BEGIN PROCEDURE SET Avg to (Mark1 + Mark2 + Mark3)/3 END PROCEDURE

Subprograms		
Syntax	Explanation of syntax	Example
<pre> FUNCTION <id> (<parameter>, ...) BEGIN FUNCTION <command> RETURN <expression> END FUNCTION </pre>	<p>Defines a function.</p>	<pre> FUNCTION AddMarks (Mark1, Mark2, Mark3) BEGIN FUNCTION SET Total to (Mark1 + Mark2 + Mark3)/3 RETURN Total END FUNCTION </pre>
<pre> <id> (<parameter>, ...) </pre>	<p>Calls a procedure or a function.</p>	<pre> Add (FirstMark, SecondMark) </pre>

Arithmetic operators	
Symbol	Description
+	Add
-	Subtract
/	Divide
*	Multiply
^	Exponent
MOD	Modulo
DIV	Integer division

Relational operators	
Symbol	Description
=	equal to
<>	not equal to
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to

Logical operators	
Symbol	Description
AND	Returns true if both conditions are true.
OR	Returns true if any of the conditions are true.
NOT	Reverses the outcome of the expression; true becomes false, false becomes true.