Pseudocode

The following information sets out how pseudocode will appear within the examinations of this syllabus. The numbers and letters that appear at the end of a sub-heading provide a cross reference to the relevant section of the subject content.

General style

Font style and size

Pseudocode is presented in `Courier New`. The size of the font will be consistent throughout.

Indentation

Lines are indented by four spaces to indicate that they are contained within a statement in a previous line. Where it is not possible to fit a statement on one line, any continuation lines are indented by two spaces from the margin. In cases where line numbering is used, this indentation may be omitted. Every effort will be made to make sure that code statements are not longer than a line of code, unless this is necessary.

Note that the `THEN` and `ELSE` clauses of an `IF` statement are indented by only two spaces. Cases in `CASE` statements are also indented by only two spaces.

Case

Keywords are in upper case, e.g. `IF`, `REPEAT`, `PROCEDURE`.

Identifiers are in mixed case with upper case letters indicating the beginning of new words, e.g. `NumberOfPlayers`.

Meta-variables – (symbols in the pseudocode that should be substituted by other symbols) are enclosed in angled brackets < >.

Example – meta-variables

```
REPEAT
    <Statements>
UNTIL <Condition>
```

Lines and line numbering

Each line representing a statement is numbered. However, when a statement runs over one line of text, the continuation lines are not numbered.

Comments

Comments are preceded by two forward slashes: `//`. The comment continues until the end of the line. For multi-line comments, each line is preceded by `//`.

Normally the comment is on a separate line before, and at the same level of indentation as, the code it refers to. Occasionally, however, a short comment that refers to a single line may be at the end of the line to which it refers.

Example – comments

```
// This procedure swaps
// values of X and Y
PROCEDURE SWAP(X : INTEGER, Y : INTEGER)
    Temp ← X    // temporarily store X
    X ← Y
    Y ← Temp
ENDPROCEDURE
```

Variables, constants and data types

Basic data types (8.1.2)

The following keywords are used to designate basic data types:

• `INTEGER`	a whole number
• `REAL`	a number capable of containing a fractional part
• `CHAR`	a single character
• `STRING`	a sequence of zero or more characters
• `BOOLEAN`	the logical values `TRUE` and `FALSE`

Literals

Literals of the above data types are written as follows:

• Integer	written as normal in the denary system, e.g. `5`, `-3`
• Real	always written with at least one digit on either side of the decimal point, zeros being added if necessary, e.g. `4.7`, `0.3`, `-4.0`, `0.0`
• Char	a single character delimited by single quotes, e.g. `'x'`, `'c'`, `'@'`
• String	delimited by double quotes. A string may contain no characters (i.e. the empty string), e.g. `"This is a string"`, `""`
• Boolean	`TRUE`, `FALSE`

Identifiers

Identifiers (the names given to variables, constants, procedures and functions) are in mixed case using Pascal case, e.g. `FirstName`. They can only contain letters (`A-Z`, `a-z`) and digits (`0-9`). They must start with a capital letter and not a digit. Accented letters and other characters, including the underscore, should not be used.

As in programming, it is good practice to use identifier names that describe the variable, procedure or function to which they refer. Single letters may be used where these are conventional (such as `i` and `j` when dealing with array indices, or `X` and `Y` when dealing with coordinates) as these are made clear by the convention.

Keywords should never be used as identifier names.

Identifiers should be considered case insensitive, for example, `Countdown` and `CountDown` should not be used as separate variables.

Variable declarations (8.1.1)

Declarations are made as follows:

```
DECLARE <identifier> : <data type>
```

Example – variable declarations

```
DECLARE Counter : INTEGER
DECLARE TotalToPay : REAL
DECLARE GameOver : BOOLEAN
```

Constants (8.1.1)

It is good practice to use constants if this makes the pseudocode more readable, and easier to update if the value of the constant changes.

Constants are declared by stating the identifier and the literal value in the following format:

```
CONSTANT <identifier> ← <value>
```

Example – `CONSTANT` declarations

```
CONSTANT HourlyRate ← 6.50
CONSTANT DefaultText ← "N/A"
```

Only literals can be used as the value of a constant. A variable, another constant or an expression must never be used.