Pseudocode

Back to basics

Content

1. Conventions
2. Common Operations
3. Selection
4. Iteration
5. Procedures & Functions
6. Arrays
7. Abstract Data Types (ADT)
8. File handling

1. Conventions

- Font: `Courier New`
- Indentation to show blocks of code (just like Python)
- Keywords are in uppercase e.g. `IF REPEAT, UNTIL, PROCEDURE, THEN, ELSE`
- Data types: `INTEGER, REAL, CHAR, STRING, BOOLEAN, DATE`
- Identifiers (variable names): use CamelCase, e.g. `StudentName, Counter`
- Declaration of variables:
  - Syntax: `DECLARE <identifier> : <data type>`
  - Example: `DECLARE Counter : INTEGER`
- Assignments: use arrow ←
  - `Counter ← 0`
  - `Counter ← Counter + 1`

2. Common Operations

- I/O: INPUT/OUTPUT

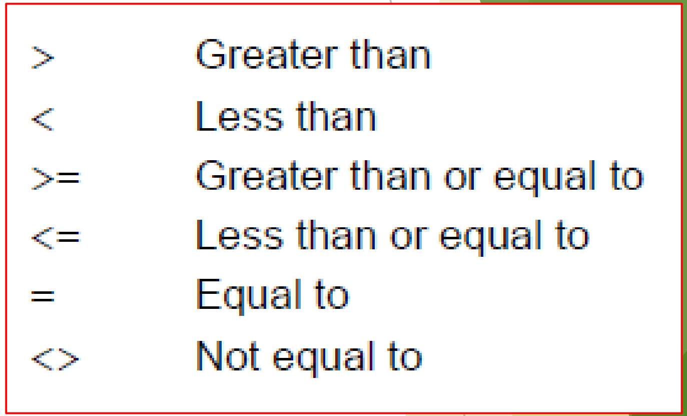
Syntax: INPUT <identifier>

Syntax: OUTPUT <value(s)>

e.g: INPUT Age

e.g: OUTPUT "Your entered age is", Age

- Arithmetic: +, -, *, /, MOD, DIV
- Relational Operations: >, <, >=, <=, =, <>
- Logic Operators: AND, OR, NOT
- Random Number Generation: RANDOMBETWEEN (min, max)
- String operations e.g. slicing, concatenation should be explained clearly.



>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
=	Equal to
<>	Not equal to

3. Conditionals/Selection

- IF statements Syntax:

```
IF <condition>
   THEN
      <statements>
ENDIF
```

- IF statements e.g.:

```
IF Age <= 12
   THEN
      Status ← ‘Child’
ENDIF
```

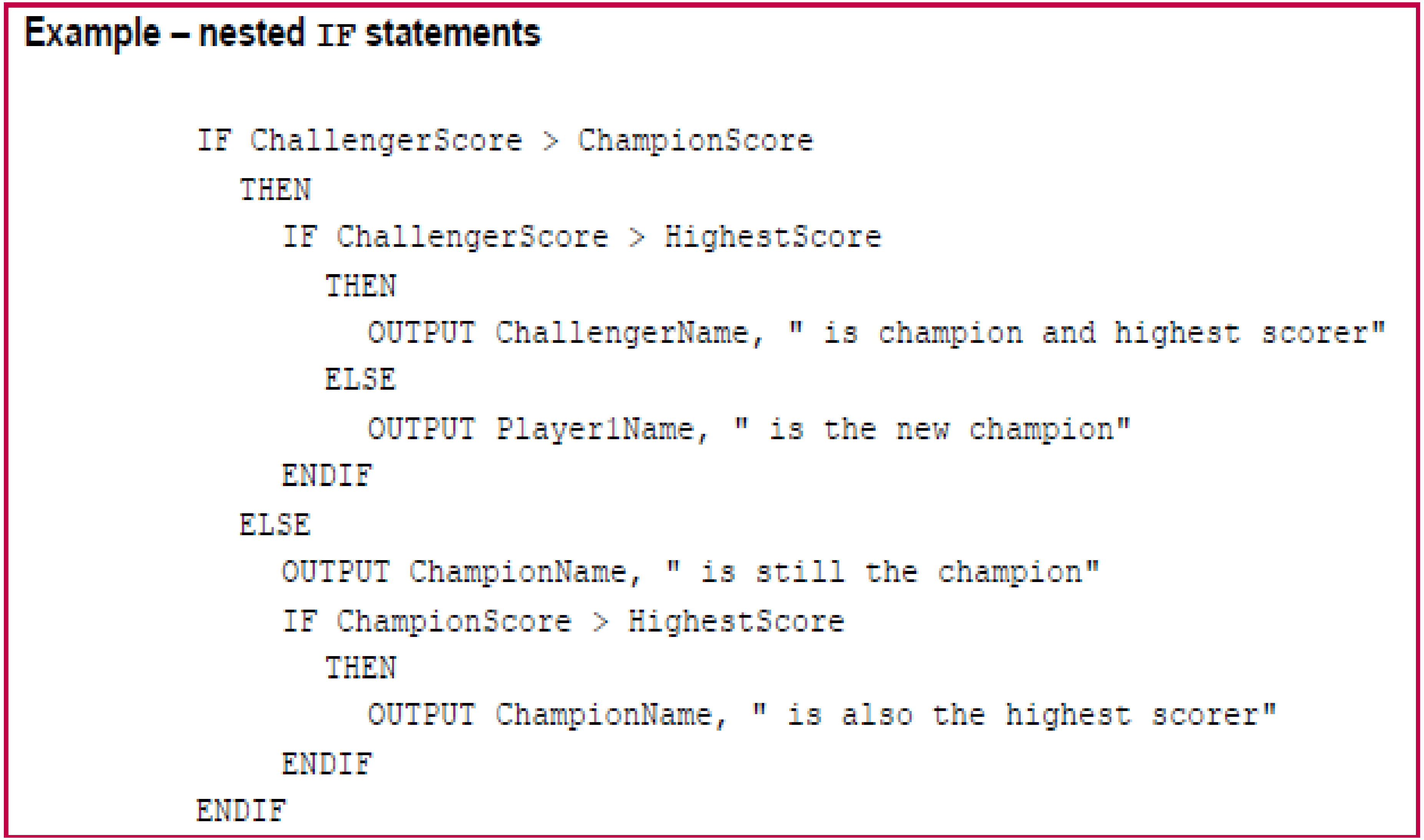
- IF/ELSE Syntax:

```
IF <condition>
   THEN
      <statements>
   ELSE
      <statements>
ENDIF
```

- IF/ELSE e.g.:

```
IF Age <= 12
   THEN
      Status ← ‘Child’
   ELSE
      Status ← ‘Adult’
ENDIF
```

3. Conditionals/Selection (Nested IF)



Example – nested `IF` statements

```
IF ChallengerScore > ChampionScore
  THEN
    IF ChallengerScore > HighestScore
      THEN
        OUTPUT ChallengerName, " is champion and highest scorer"
      ELSE
        OUTPUT Player1Name, " is the new champion"
    ENDIF
  ELSE
    OUTPUT ChampionName, " is still the champion"
    IF ChampionScore > HighestScore
      THEN
        OUTPUT ChampionName, " is also the highest scorer"
    ENDIF
ENDIF
```

4. Iteration (Count-controlled (FOR) loops)

- Count-controlled (FOR) loops syntax:

```
FOR <identifier> ← <value1> TO <value2>
    <statements>
ENDFOR
```

- Count-controlled (FOR) loops e.g.:

```
FOR i ← 1 TO 10
    Sum ← Sum + i
ENDFOR
```

4. Iteration (Post-condition (REPEAT UNTIL) loops):

- Post-condition (REPEAT UNTIL) loops syntax:

```
REPEAT
   <Statements>
UNTIL <condition>
```

- Post-condition (REPEAT UNTIL) loops e.g.:

```
REPEAT
   OUTPUT "Please enter the password"
   INPUT Password
UNTIL Password = "Secret"
```

4. Iteration (Pre-condition (WHILE) loops):

- Pre-condition (WHILE) loops syntax:

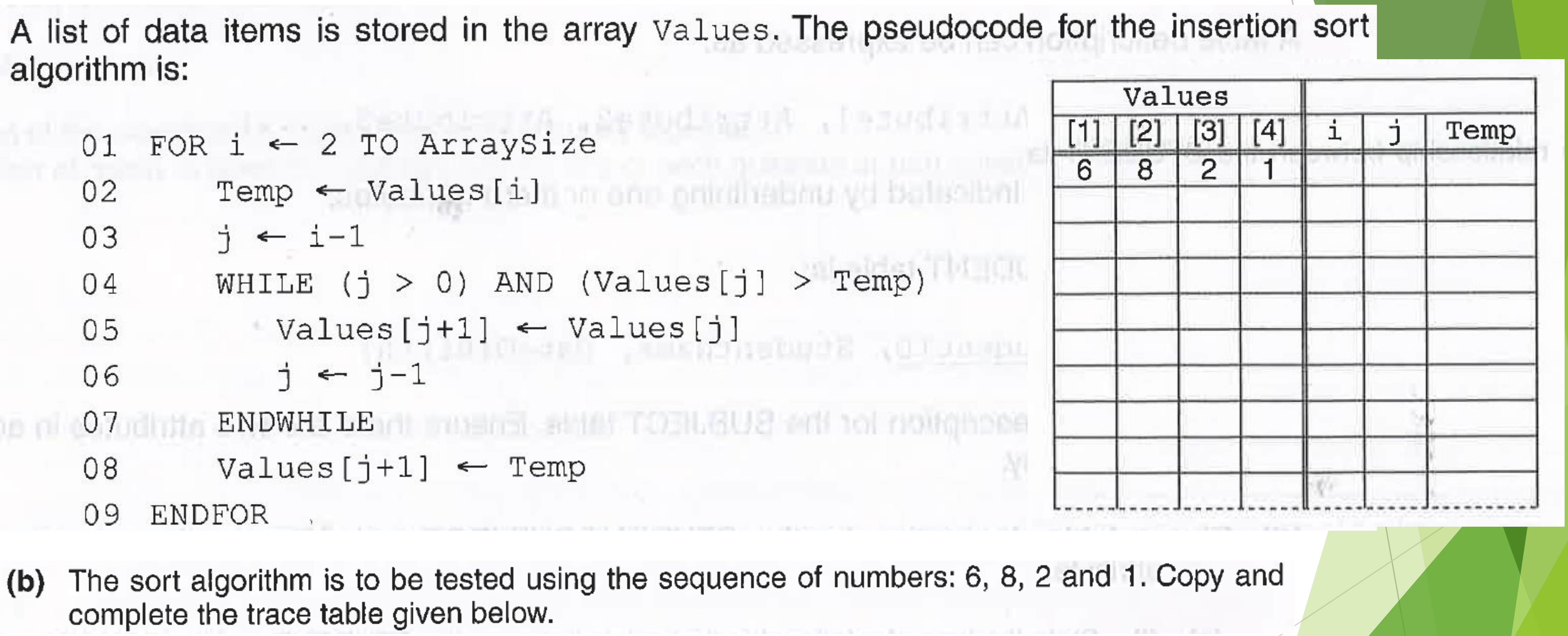
```
WHILE <condition> DO
   <Statements>
ENDWHILE
```

- Pre-condition (WHILE) loops e.g.:

```
WHILE Number > 9 DO
   Number ← Number - 9
ENDWHILE
```

A level 2015 P2Q5

Assignment 1



A list of data items is stored in the array Values. The pseudocode for the insertion sort algorithm is:

```
FOR i ← 2 TO ArraySize
    Temp ← Values[i]
    j ← i-1
    WHILE (j > 0) AND (Values[j] > Temp)
      Values[j+1] ← Values[j]
      j ← j-1
    ENDWHILE
    Values[j+1] ← Temp
ENDFOR
```

Values						
[1]	[2]	[3]	[4]	i	j	Temp
6	8	2	1			

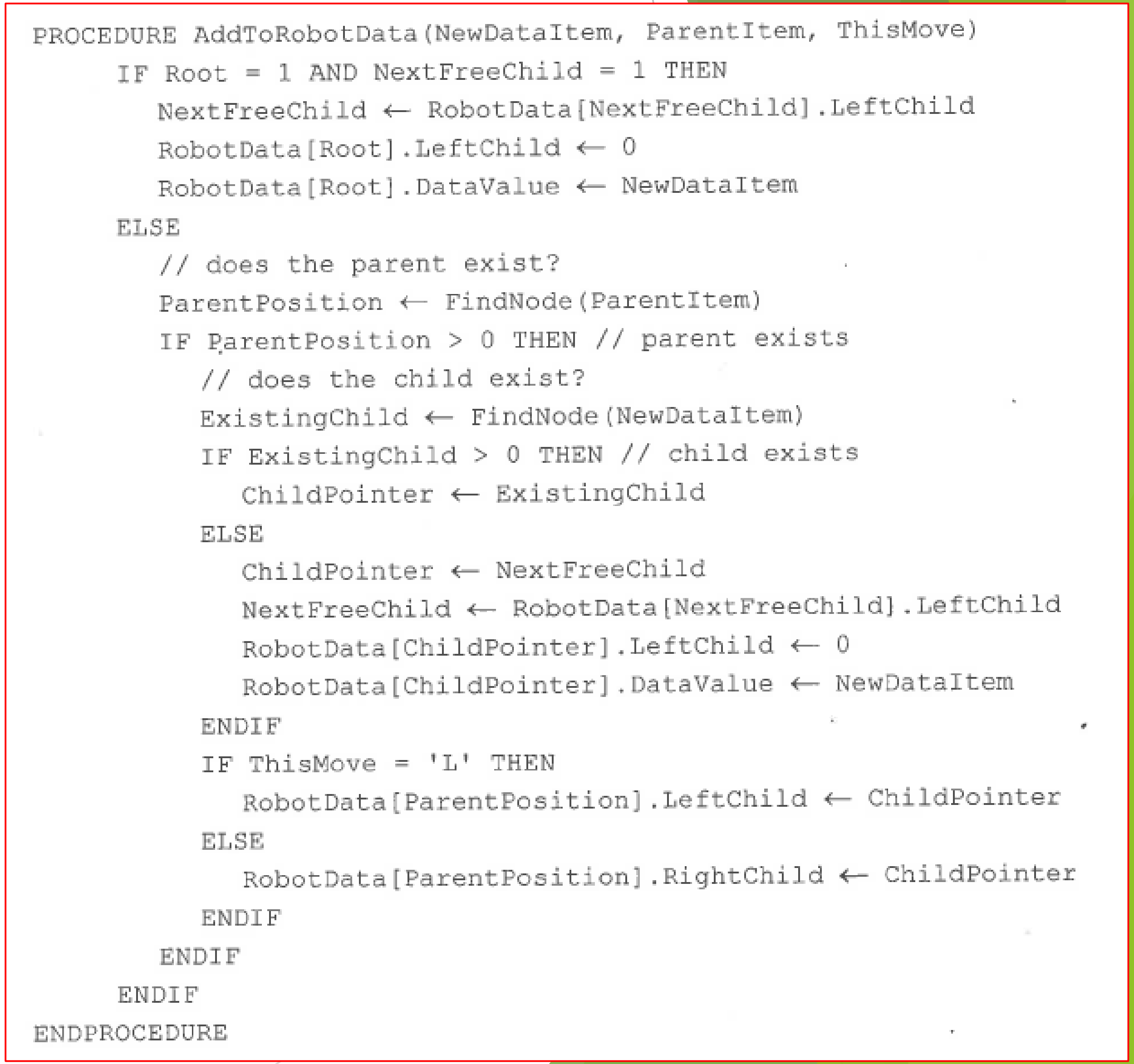
(b) The sort algorithm is to be tested using the sequence of numbers: 6, 8, 2 and 1. Copy and complete the trace table given below.

5. Procedures & Functions

- Defining procedures, syntax:

```
PROCEDURE <identifier> (Parameters)
   <statements>
ENDPROCEDURE
```

- Defining procedures, e.g.:



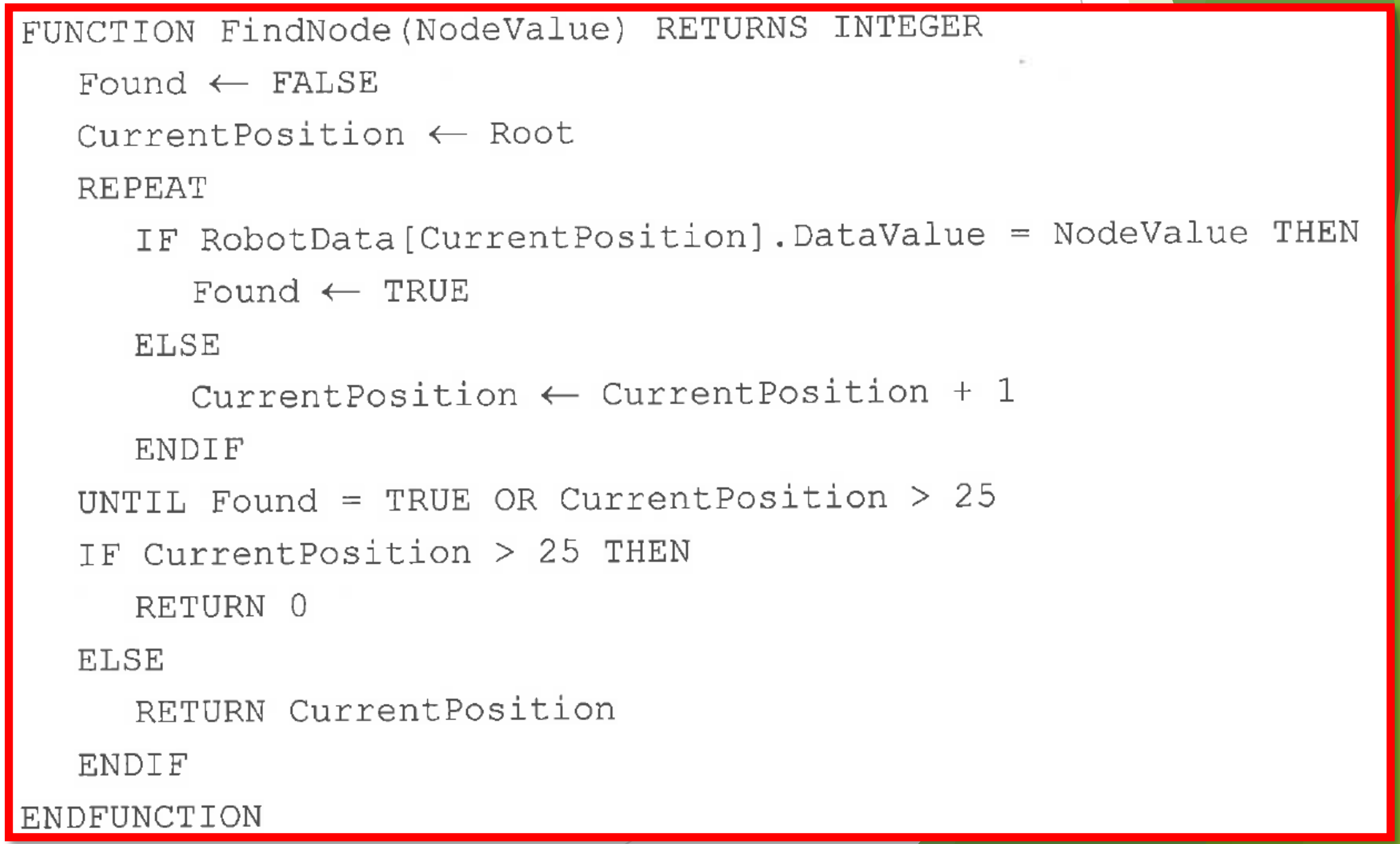
```
PROCEDURE AddToRobotData(NewDataItem, ParentItem, ThisMove)
    IF Root = 1 AND NextFreeChild = 1 THEN
      NextFreeChild ← RobotData[NextFreeChild].LeftChild
      RobotData[Root].LeftChild ← 0
      RobotData[Root].DataValue ← NewDataItem
    ELSE
      // does the parent exist?
      ParentPosition ← FindNode(ParentItem)
      IF ParentPosition > 0 THEN // parent exists
        // does the child exist?
        ExistingChild ← FindNode(NewDataItem)
        IF ExistingChild > 0 THEN // child exists
          ChildPointer ← ExistingChild
        ELSE
          ChildPointer ← NextFreeChild
          NextFreeChild ← RobotData[NextFreeChild].LeftChild
          RobotData[ChildPointer].LeftChild ← 0
          RobotData[ChildPointer].DataValue ← NewDataItem
        ENDIF
        IF ThisMove = 'L' THEN
          RobotData[ParentPosition].LeftChild ← ChildPointer
        ELSE
          RobotData[ParentPosition].RightChild ← ChildPointer
        ENDIF
      ENDIF
    ENDIF
ENDPROCEDURE
```

5. Procedures & Functions

- Defining functions, syntax:

```
FUNCTION <identifier> RETURNS <data type>
   <statements>
ENDFUNCTION
```

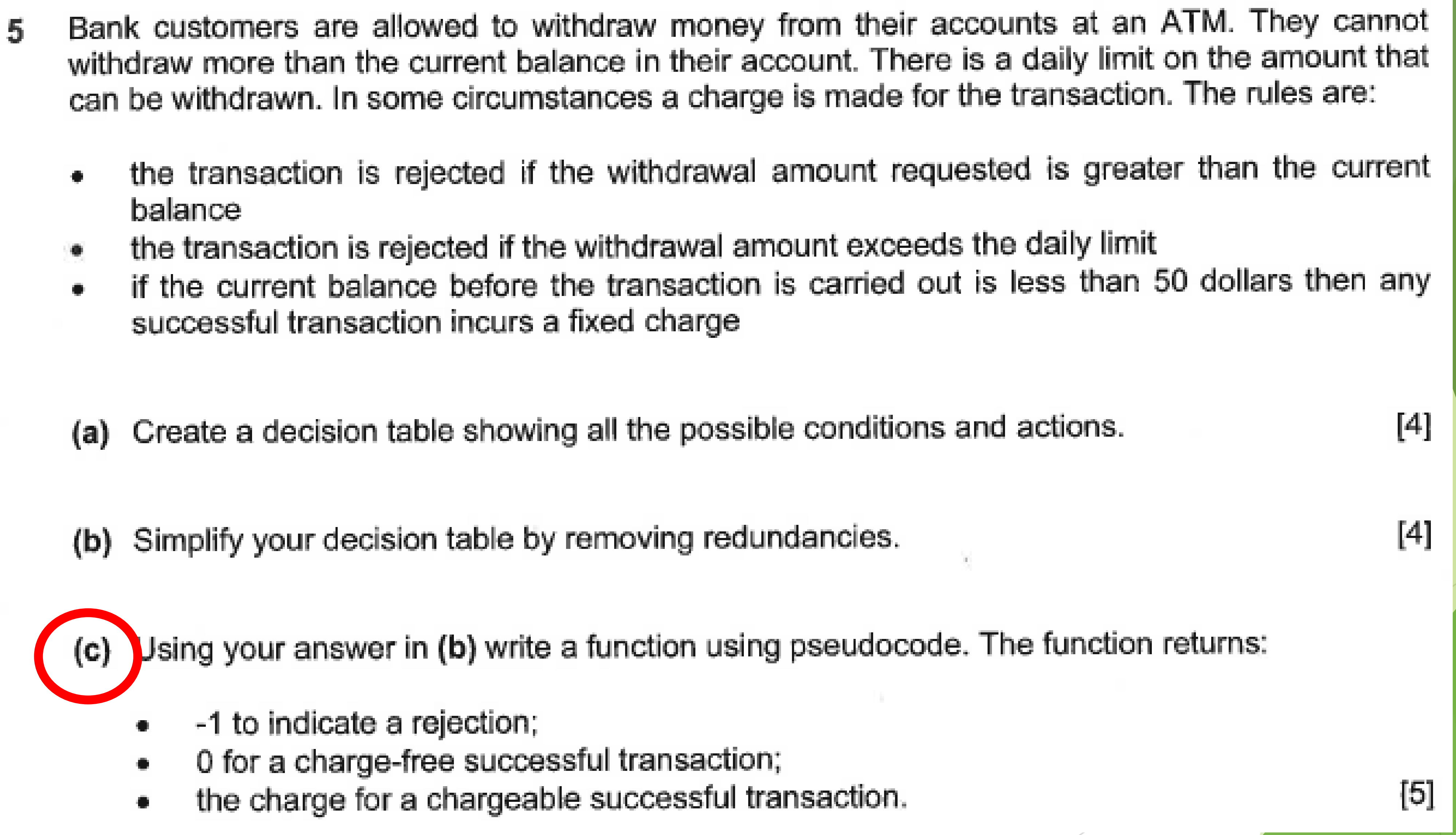
- Defining functions, e.g.:



```
FUNCTION FindNode(NodeValue) RETURNS INTEGER
   Found ← FALSE
   CurrentPosition ← Root
   REPEAT
      IF RobotData[CurrentPosition].DataValue = NodeValue THEN
         Found ← TRUE
      ELSE
         CurrentPosition ← CurrentPosition + 1
      ENDIF
   UNTIL Found = TRUE OR CurrentPosition > 25
   IF CurrentPosition > 25 THEN
      RETURN 0
   ELSE
      RETURN CurrentPosition
   ENDIF
ENDFUNCTION
```

A level 2013 P2Q5

Assignment 2



5 Bank customers are allowed to withdraw money from their accounts at an ATM. They cannot withdraw more than the current balance in their account. There is a daily limit on the amount that can be withdrawn. In some circumstances a charge is made for the transaction. The rules are:

- the transaction is rejected if the withdrawal amount requested is greater than the current balance
- the transaction is rejected if the withdrawal amount exceeds the daily limit
- if the current balance before the transaction is carried out is less than 50 dollars then any successful transaction incurs a fixed charge

(a) Create a decision table showing all the possible conditions and actions. [4]

(b) Simplify your decision table by removing redundancies. [4]

(c) Using your answer in **(b)** write a function using pseudocode. The function returns:

- -1 to indicate a rejection;
- 0 for a charge-free successful transaction;
- the charge for a chargeable successful transaction. [5]

6. Arrays

- Fixed-length data structures, containing elements of identical data types
- Elements accessible by index number, using index operator [ ]
- Lower bound (index of first element) either 0 or 1
- 1-dimensional array:

```
[<elem1>, <elem2>, <elem3>]
```

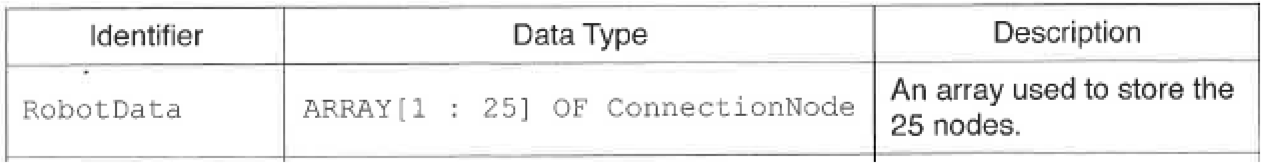
- 2-dimensional arrays:

```
[[<ele11>,<ele12>,<ele13>],[<ele21>,<ele22>,<ele23>],
[<ele31>,<ele32>, <ele33>]]
```

- To access an element in a 1-d array, one index number is sufficient, whereas for 2-d array, 2 indices must be specified for the row and column.

6. Arrays

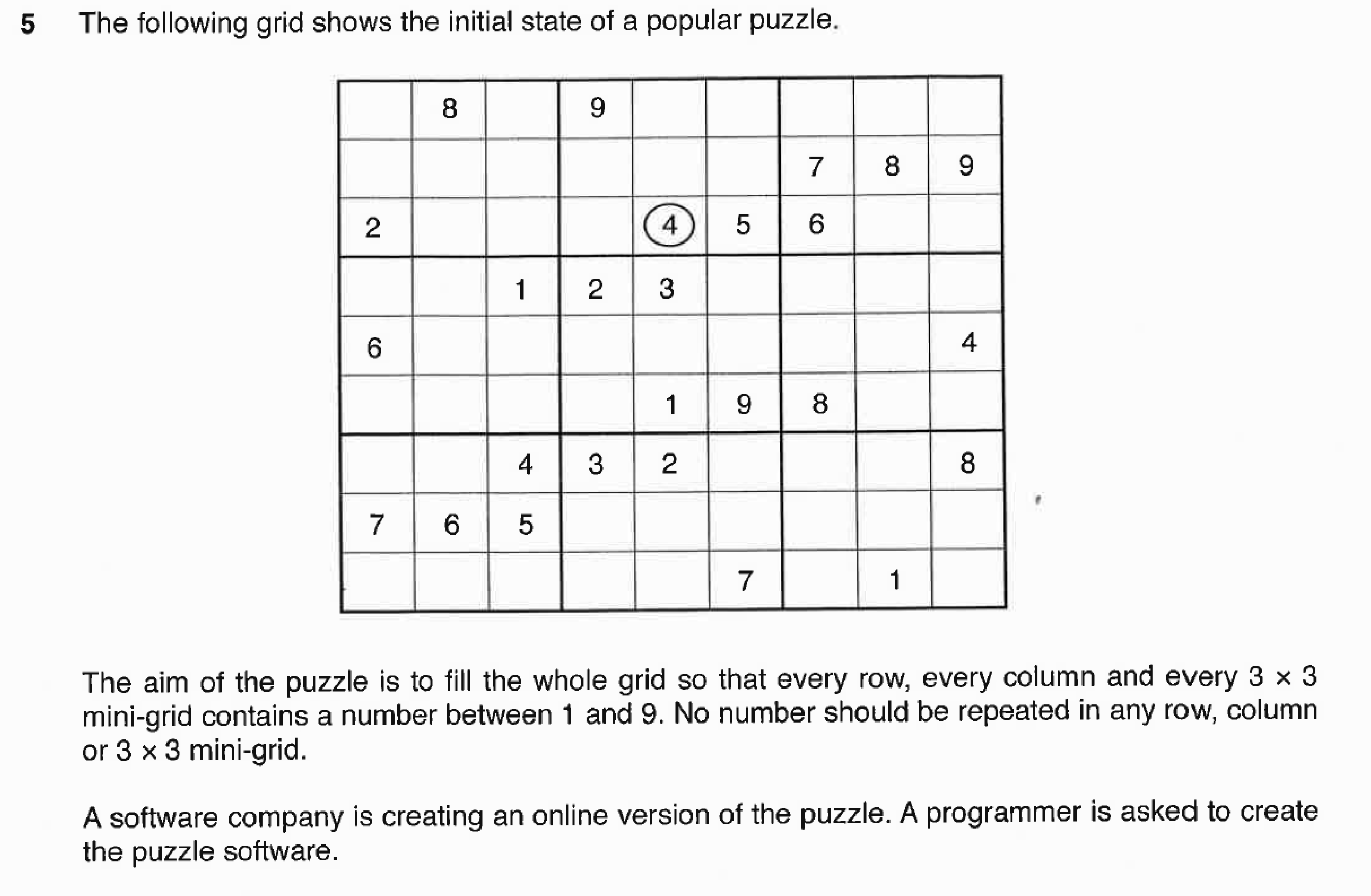
- Declaration of array:
  - 1-D syntax :
  - DECLARE <identifier> : ARRAY [<l>:<u>] OF <data type>
  - e.g. DECLARE StudentName: ARRAY [1:30] OF STRING



Identifier	Data Type	Description
RobotData	ARRAY[1 : 25] OF ConnectionNode	An array used to store the 25 nodes.

  - 2-D syntax :
  - DECLARE <identifier> : ARRAY [<l1>:<u1>, <l2>:<u2>] OF <data type>
  - e.g. DECLARE TicTacToe: ARRAY [1:3, 1:3] OF CHAR
- Assignments in Arrays: ←
  - StudentNames[1] ← “Ali”
  - TicTacToe[2,3] ← ‘X’

A level 2017 P2Q5



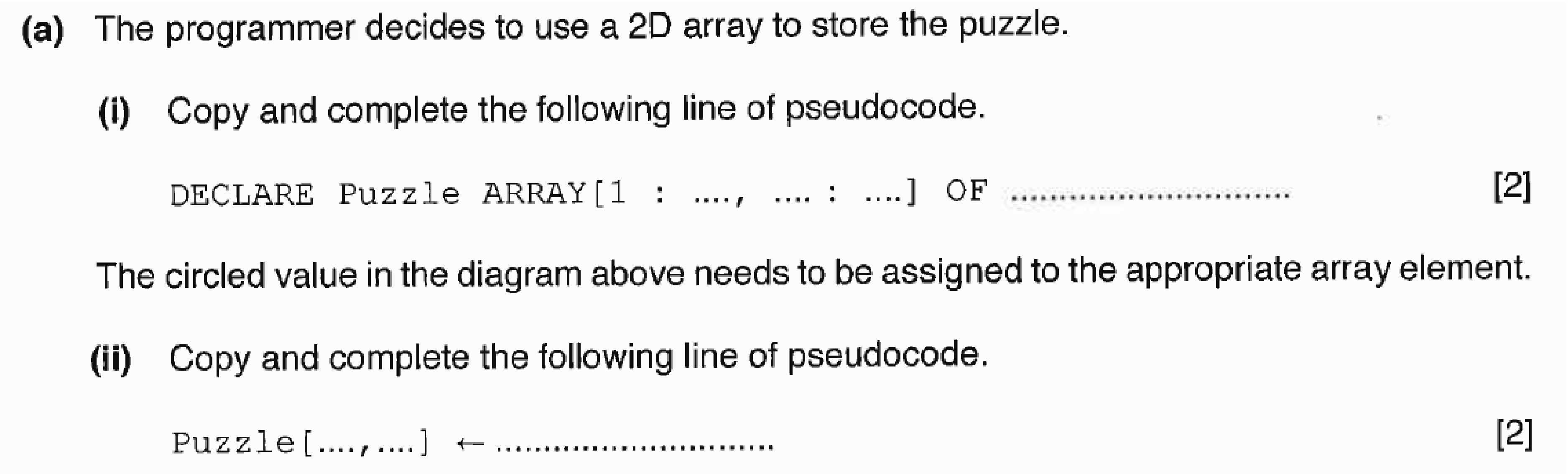
5 The following grid shows the initial state of a popular puzzle.

	8		9					
						7	8	9
2				④	5	6		
		1	2	3				
6								4
				1	9	8		
		4	3	2				8
7	6	5						
					7		1	

The aim of the puzzle is to fill the whole grid so that every row, every column and every 3 × 3 mini-grid contains a number between 1 and 9. No number should be repeated in any row, column or 3 × 3 mini-grid.

A software company is creating an online version of the puzzle. A programmer is asked to create the puzzle software.

A level 2017 P2Q5



(a) The programmer decides to use a 2D array to store the puzzle.

(i) Copy and complete the following line of pseudocode.

```
DECLARE Puzzle ARRAY[1 : ...., .... : ....] OF ..............................
```

[2]

The circled value in the diagram above needs to be assigned to the appropriate array element.

(ii) Copy and complete the following line of pseudocode.

```
Puzzle[....,....] ← ..............................
```

[2]

7. Abstract Data Type (ADT)

- Custom data structures that are not available in a particular programming language need to be constructed from the data structures that are built-in within the language.
- e.g. Queue, Stack and their associated constructor, getters and setters using OOP
- Declaration of ADT, syntax:

```
TYPE <identifier>
    DECLARE <attribute1>: <data type>
    DECLARE <attribute2>: <data type>
    DECLARE <attribute3>: <data type>
    ...
ENDTYPE
```

Declaration of ADT, e.g.:

```
TYPE Student
    DECLARE Surname: STRING
    DECLARE FirstName: STRING
    DECLARE DateOfBirth : DATE
    DECLARE YearGroup: INTEGER
    DECLARE CivicsGroup: STRING
ENDTYPE
```

7. Abstract Data Type (ADT)

```
TYPE Student
    DECLARE Surname: STRING
    DECLARE FirstName: STRING
    DECLARE DateOfBirth : DATE
    DECLARE YearGroup: INTEGER
    DECLARE CivicsGroup: STRING
ENDTYPE
```

```
DECLARE Pupil1: Student
DECLARE Pupil2: Student
Pupil1.Surname ← "John"
Pupil1.Firstname ← "Leroy"
Pupil1.DateOfBirth ← 02/01/2005
Pupil1.YearGroup ← 2
Pupil1.CivicsGroup ← "CTG245"
```

```
DECLARE Form: ARRAY[1:30] OF Student
FOR Index ← 1 TO 30
   Form[Index].YearGroup ← Form[Index].YearGroup + 1
ENDFOR
```

8. File handling

- Opening a file syntax:

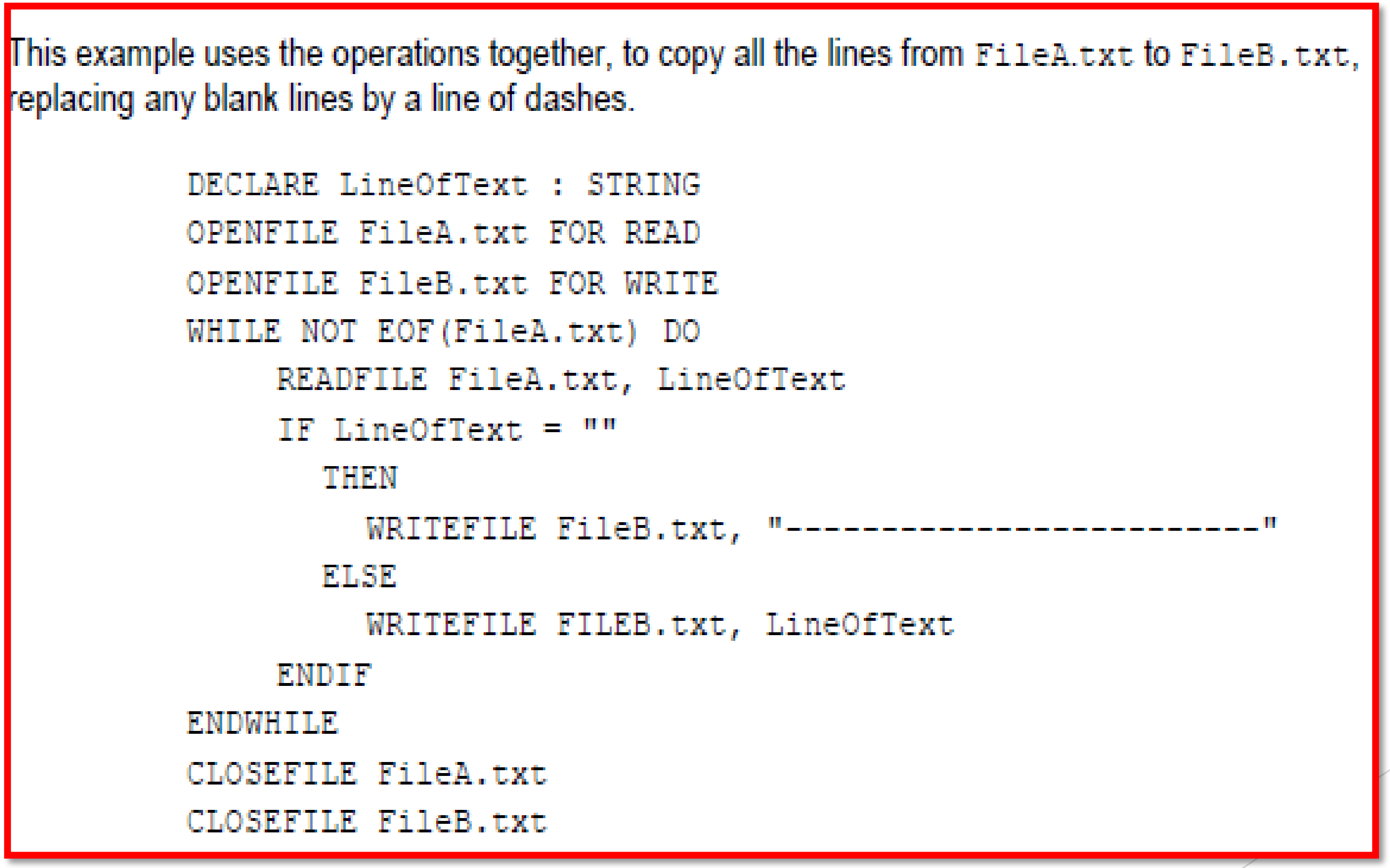
```
OPENFILE <File identifier> FOR <File mode>
```

- `File mode: READ, WRITE, APPEND`
- Opening a file e.g.:

```
OPENFILE ‘STUDENT.TXT’ FOR READ
```

- If READ:
  - `READFILE <File identifier> , <Variable>`
- If WRITE:
  - `WRITEFILE <File identifier> , <String>`
- Closing a file:
  - `CLOSEFILE <File identifier>`

8. File handling

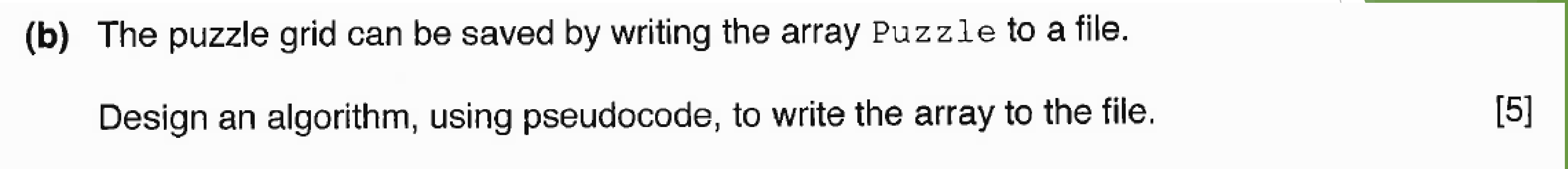


This example uses the operations together, to copy all the lines from `FileA.txt` to `FileB.txt`, replacing any blank lines by a line of dashes.

```
DECLARE LineOfText : STRING
OPENFILE FileA.txt FOR READ
OPENFILE FileB.txt FOR WRITE
WHILE NOT EOF(FileA.txt) DO
     READFILE FileA.txt, LineOfText
     IF LineOfText = ""
       THEN
         WRITEFILE FileB.txt, "-------------------------"
       ELSE
         WRITEFILE FILEB.txt, LineOfText
     ENDIF
ENDWHILE
CLOSEFILE FileA.txt
CLOSEFILE FileB.txt
```

A level 2017 P2Q5

Assignment 3



(b) The puzzle grid can be saved by writing the array `Puzzle` to a file.

Design an algorithm, using pseudocode, to write the array to the file. [5]