Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

COMPUTER SCIENCE **9608/02**

Paper 2 Fundamental Problem-solving and Programming Skills **For Examination from 2015**

SPECIMEN MARK SCHEME

2 hours

MAXIMUM MARK: 75

This document consists of **7** printed pages and **1** blank page.

CAMBRIDGE International Examinations

1

```
Dim HomeTeamName As String
Dim AwayTeamName As String
Dim WinningTeamName As String

Dim HomeRuns As Integer
Dim AwayRuns As Integer
Dim RunDifference As Integer

HomeTeamName = Console.ReadLine
HomeRuns = Console.ReadLine
AwayTeamName = Console.ReadLine
AwayRuns = Console.ReadLine

If HomeRuns > AwayRuns Then
   WinningTeamName = HomeTeamName
Else
   WinningTeamName = AwayTeamName
End If

RunDifference = Math.Abs(HomeRuns - AwayRuns)

Console.WriteLine("Winning team was " & WinningTeamName
   & " who scored " & RunDifference & " more runs")
```

Mark as follows:

Declaration of name strings	[1]
Declaration of scores	[1]
Input for name strings	[1]
Input of two scores	[1]
Calculation of the runs difference	[1]
Calculation of the difference	[1]
2 × IF or IF-THEN-ELSE used	[1]
Stored as WinningTeamName	[1]
Output shows team and runs difference	[1]

[Total: 9]

2 **(a)** **(i)** *Identifier table:*
INTEGER [1]
Explanation – the next number selected [1]

(ii) *Pseudocode:*

```
FOR Counter ←1 to 6
   NextNumber ← INT(RND()*50) + 1          [1]
   OUTPUT NextNumber                       [1]
ENDFOR / anything to mark the end of the loop [1]
OUTPUT "That completes the draw"
```

(b) Program code demonstrates:
declaration of variables [1]
correctly formed 'count-controlled' loop [1]
clear use of relevant inbuilt function [1]

(c) **(i)** Explanation, e.g., It is not known how many times the loop needs to be executed to generate 6 different numbers. [1]

(ii) any post-condition or pre-condition loop [1]

(iii)

```
PROCEDURE InitialiseNumberDrawn
   FOR Index ← 1 TO 50
      NumberDrawn[Index] ← FALSE
   ENDFOR                                  [3]
END PROCEDURE
```

(iv)

```
CALL InitialiseNumberDrawn
Generated ← 0
REPEAT // start of loop
   NextNumber ← GenerateNumber()
   IF NumberDrawn[NextNumber] = FALSE      [2]
      THEN
         OUTPUT NextNumber
         Generated ← Generated + 1         [1]
         NumberDrawn[NextNumber] ← TRUE
   ENDIF                                   [2]
UNTIL Generated = 6 // end of loop         [1]
OUPUT "That completes the draw"
```

(v)

NumberDrawn	
1	FALSE
2	FALSE
3	TRUE
4	FALSE
5	FALSE
6	FALSE
7	FALSE
8	FALSE
9	TRUE
10	FALSE
…	∫
39	FALSE
40	FALSE
41	FALSE
42	TRUE
43	FALSE
44	FALSE
45	FALSE
46	FALSE
47	TRUE
48	FALSE
49	FALSE
50	FALSE

Mark as follows:

4 × correct 'TRUE' cells [1]
All other cells FALSE [1]
All cells contain something [1]

(vi) 3 47 9 42 [1]

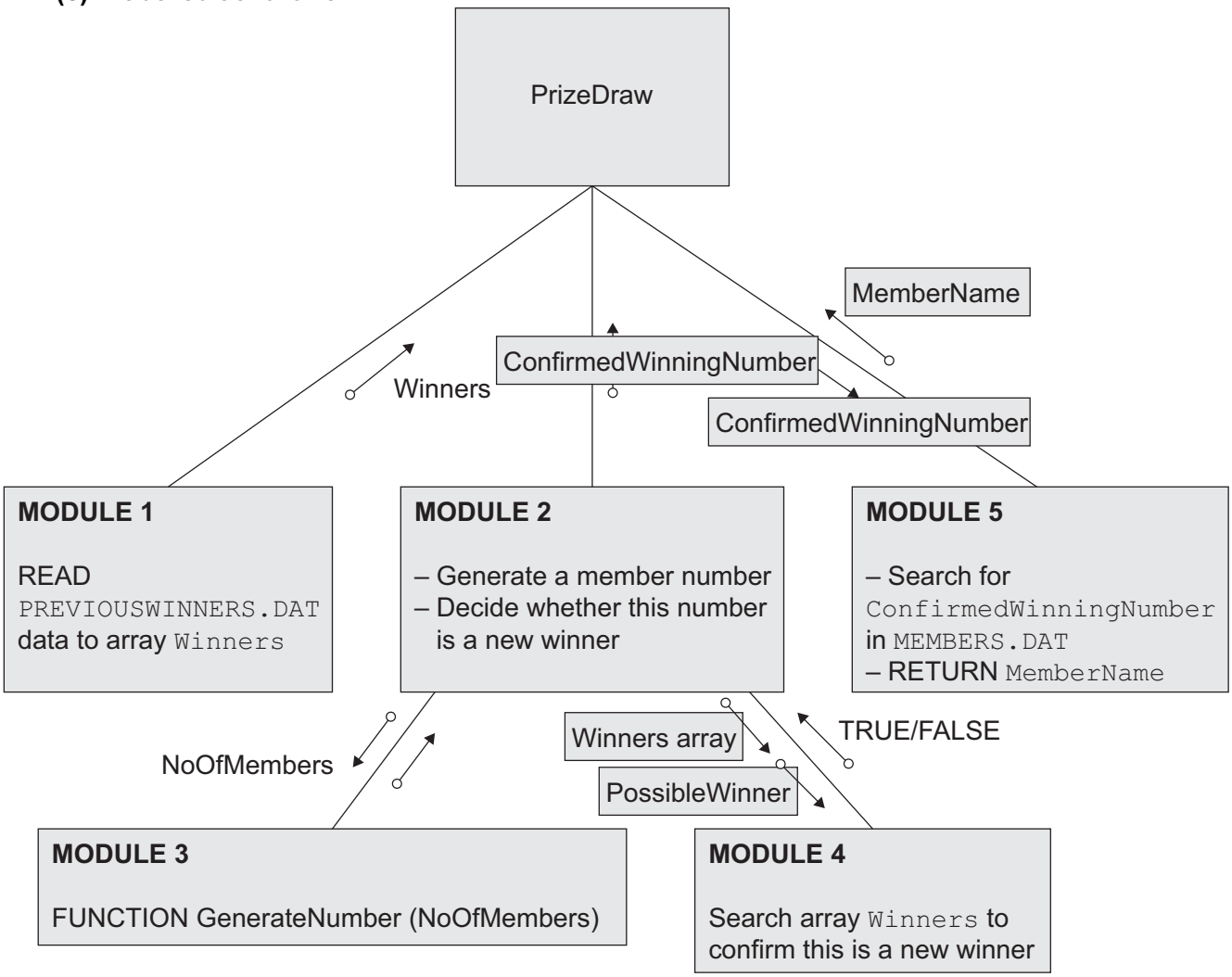
[Total: 23]

3 **(a)** **(i)** 1 the identifier name for the function (chosen by the programmer) [1]
2 the parameter [1]
3 data type (for the parameter) [1]
4 data type for the value <u>returned</u> by the function [1]

(ii) Variable `PossibleWinner` stores the value returned by the function. [1]

(b) The data must be available each week. [1]
When the program terminates after each weekly run, the data must be saved. [1]

(c) Labelled as follows:



[6]

(d) **(i)** `Index- INTEGER` – Array subscript [3]

(ii) *Mark as follows:*

procedure header	[1]
open the file	[1]
correct open mode used	[1]
index initialised	[1]
loop	[1]
read line of text	[1]
assign to next array element	[1]
increment index	[1]
test for EOF	[1]
output message shown	[1]
	[max 8]

(e) **(i)** `DataLength ← LEN(MemberData)` [1]

(ii) `MemberNumber ← LEFT(MemberData, 4)` [1]

(iii) `MemberName ← MID(MemberData, 6, DataLength - 5)` [1]

[Total: 27]

4 **(a)** **(i)** P [1]

(ii) 87 [1]

(b) 84 [1]

(c) `PEKOHOX` [1]

(d) (i)

```
INPUT MessageString
LengthMessageString ← LEN(MessageString)
NewString ← ""
FOR CharacterPosition ← 1 TO LengthMessageString
   Found ← FALSE
   Index ← 1
   REPEAT
      IF MessageString[CharacterPosition] = Alphabet[Index]
         THEN
            SubstituteCharacter ← Substitute[Index]
            Found ← TRUE
      ELSE
         Index ← Index + 1
      ENDIF
   UNTIL Found
   NewString ← NewString + SubstituteCharacter
ENDFOR
OUTPUT NewString
```

Mark as follows:

input of the string	[1]
assign `NewString` as empty	[1]
calculation of the string length	[1]
outer loop	[1]
for 'length' iterations	[1]
compare individual characters with `Alphabet` array	[1]
inner loop to search for character	[1]
controlled with a counter	[1]
new substitute character added to `NewString`	[1]
final output of `NewString`	[1]
	[max 10]

(ii) The code to search the Alphabet array can be avoided. / The ASCII codes for the letters are in sequence.

Example – index position for any character is `ASC(<char>)-64` [2]

[Total: 16]

BLANK PAGE