CAMBRIDGE INTERNATIONAL EXAMINATIONS

Cambridge International Advanced Subsidiary and Advanced Level

MARK SCHEME for the October/November 2014 series

9691 COMPUTING

9691/23 Paper 2 (Written Paper), maximum raw mark 75

This mark scheme is published as an aid to teachers and candidates, to indicate the requirements of the examination. It shows the basis on which Examiners were instructed to award marks. It does not indicate the details of the discussions that took place at an Examiners' meeting before marking began, which would have considered the acceptability of alternative answers.

Mark schemes should be read in conjunction with the question paper and the Principal Examiner Report for Teachers.

Cambridge will not enter into discussions about these mark schemes.

Cambridge is publishing the mark schemes for the October/November 2014 series for most Cambridge IGCSE®, Cambridge International A and AS Level components and some Cambridge O Level components.

CAMBRIDGE
International Examinations

1 **(a)**
```
DIM Tally[1 : 4] OF INTEGER
```

1 mark for size
1 mark for data type (must be integer) **[2]**

(b) **(i)**
```
PROCEDURE InitialiseArrayCounts
   DECLARE i : INTEGER
   FOR i ← 1 TO 4
     Tally[i] ← 0
   ENDFOR
ENDPROCEDURE
```

1 mark for declaration/local variable
1 mark for loop 1 to 4
1 mark for array element set to 0

```
PROCEDURE InputStudentChoices
    REPEAT
       INPUT Choice
       Tally[Choice] ← Tally[Choice] + 1
    UNTIL Choice = 0
ENDPROCEDURE
```

1 mark for replacing CASE statement with single array element assignment **[4]**

(ii) `Football` **[1]**
Accept f.t. from (b)(i) (if array elements not numbered 1 to 4)

(c)
```
PROCEDURE OutputTallyChart
   OUTPUT "1   Cricket  "
   OutputTally(Tally[1])
   OUTPUT "2  Football  "
   OutputTally(Tally[2])
   OUTPUT "3    Tennis  "
   OutputTally(Tally[3])
   OUTPUT "4  Swimming  "
   OutputTally(Tally[4])
ENDPROCEDURE
```

2 marks for all 4 array elements correct. 1 mark for 3 correct.

```
PROCEDURE OutputTally(SportCount : INTEGER)
   IF SportCount > 0                      // 1 mark
      THEN
         FOR i ← 1 TO SportCount          // 1 mark
            OUTPUT '\'
         ENDFOR                           // 1 mark
   ENDIF
   OUTPUT NEWLINE                         // 1 mark
ENDPROCEDURE
```

[6]

(d)

Type of test data	Example test data	Justification
Normal	e.g. 1 or greater	Check correct number of bars output
Boundary	0	0 is smallest possible value And no bars should be output
Extreme	e.g. 2000	How is the procedure going to deal with a large number, more than bars fit on a line

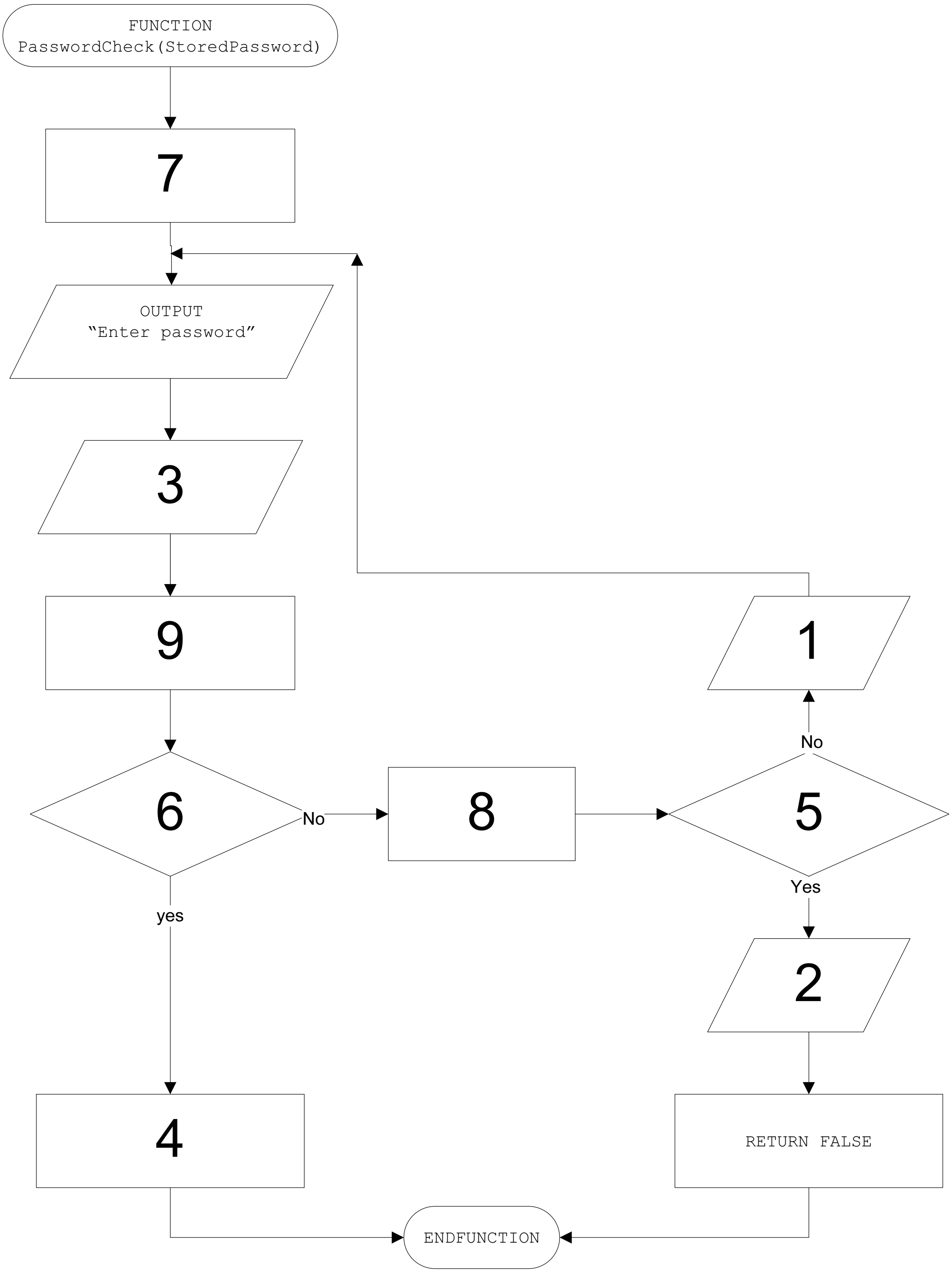
1 mark for each cell

[9]

(e) (i) – indentation
– meaningful identifiers
– initialising variables
– annotation
– parameters
– capitalisation of keywords
– modular structure **[3]**

(ii) – declaring variables/constants
– local variables **[1]**

2 (a)



FUNCTION
PasswordCheck(StoredPassword)
7
OUTPUT
"Enter password"
3
9
1
No
6
No
8
5
yes
Yes
2
4
RETURN FALSE
ENDFUNCTION

[9]

(b)

```
FUNCTION FindPassword(ThisUserID : STRING) RETURNS STRING
                                                     // 1 mark
   DECLARE Found : BOOLEAN
   OPENFILE FOR INPUT
   Found ← FALSE
   WHILE NOT EOF AND Found = FALSE                  // 2 marks
      FILEREAD next record
      IF UserID = ThisUserID                        // 1 mark
         THEN
            Found ← TRUE                            // 1 mark
      ENDIF
   ENDWHILE
   IF Found = TRUE                          // 1 mark
      THEN
         RETURN EncryptedPassword           // 1 mark
      ELSE
         RETURN Error code                          // 1 mark
   ENDIF
   CLOSEFILE
ENDFUNCTION
```

[8]

Alternative part:

```
IF Found = False                        // 1 mark
      THEN
         RETURN Error code              // 1 mark
      ELSE
         RETURN EncryptedPassword       // 1 mark
   ENDIF
```

3 **(a)** **(i)** Mark as follows:
1 mark for a heading
1 mark for input boxes with units
1 mark for text output box with description
1 mark for button "calculate" or similar
Accept console mode design **[4]**

(ii) 1 mark for explanation that fits design of (a)(i). **[1]**

(b) `(RoomWidth >=100) AND (RoomWidth < 1000)`
1 mark for each bracketed part
1 mark for AND **[3]**

(c) **(i)** 3 **[1]**

(ii) `RoomWidth MOD 30 > 0  //  RoomWidth MOD 30 != 0` **[1]**

(iii) e.g. Pascal

```
TilesForWidth := RoomWidth DIV 30;
IF RoomWidth MOD 30 > 0
   THEN TilesForWidth := TilesForWidth + 1;
TilesForLength := RoomLength DIV 30;
IF RoomLength MOD 30 > 0
   THEN TilesForLength := TilesForLength + 1;
TilesRequired := TilesForWidth * TilesForLength * 1.1; // +10%
```

1 mark for calculating tiles for length
1 mark for calculating tiles for width
1 mark for rounding up when needed
1 mark for multiplying TilesForWidth and TilesForLength
1 mark for adding 10% of total tiles required **[5]**

4 (a) (i) for example: 0 & -1 // TRUE & FALSE // `'O'` & `'X'` **[1]**

(ii) e.g. Pascal

```
VAR FloorDesign: ARRAY[1..35, 1..35] OF CHAR;    // 2 marks
```

(1 mark for correct dimensions, 1 mark for data type to match assignment below)

```
FOR i := 1 TO 35 DO                                 // 1 mark
   FOR j := 1 TO 35 DO                              // 1 mark
      FloorDesign[i,j] := 'O';                    // 1 mark
```

[5]

(b)

```
NumberOFWhiteTiles ← 0
NumberOfColourTiles ← 0
FOR a ← 1 TO 15
   FOR b ← 1 TO 10
      IF FloorDesign[a,b] = 'X'
         THEN
            NumberOfColourTiles ← NumberOfColourTiles + 1
         ELSE
            NumberOfWhiteTiles ← NumberOfWhiteTiles + 1
      ENDIF
   ENDFOR
ENDFOR
```

Mark as follows:
1 mark for initialisation
1 mark for loops with correct ranges
1 mark for correct nesting
1 mark for testing array element
1 mark for updating count of coloured tiles
1 mark for calculating number of white tiles (counting or subtracting)

[6]

5 (a)

a	x	a >= x
13		
	8	
		TRUE
5		
	4	
		TRUE
1		
	2	
		FALSE
	1	
		TRUE
0		
	0.5	

Output: 1 1 0 1

1 mark for each correct column

1 mark for correct output, in this order. [4]

(b) converts denary number to binary // converts 13 to binary [1]