CAMBRIDGE INTERNATIONAL EXAMINATIONS

Cambridge International Advanced Subsidiary and Advanced Level

MARK SCHEME for the May/June 2015 series

9691 COMPUTING

9691/22   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 May/June 2015 series for most Cambridge IGCSE®, Cambridge International A and AS Level components and some Cambridge O Level components.

1   (a) (i) `'F'`

     (ii) Error

   (b) ```
ThisDay   ← MID(TodaysDate, 1, 2)
ThisMonth ← MID(TodaysDate, 3,2)
ThisYear ← MID(TodaysDate,5,4)
```
                                                                            [3]

2   (a)  (i)

| x | Result | x < > -1 |
|---|---|---|
| 0 | 0 | TRUE |
| 3 | 3 | TRUE |
| 5 | 8 | TRUE |
| 2 | 10 | TRUE |
| 1 | 11 | TRUE |
| -1 | 10 | FALSE |

**1 mark per correct column [3]**

         OUTPUT: 10                                                          **[1]**

     (ii)  Expected result: 11                                              **[1]**

     (iii) The –1 is treated as though it was part of the sequence of numbers
           // the dummy value is included in the calculation                 **[1]**

     (iv)  Logic (error) (Accept logical)

                                                                            **[1]**

**(b)**
```
DECLARE x : INTEGER
DECLARE Result : INTEGER
x ← 0
Result ← 0
INPUT x
WHILE x <> -1
Result ← Result + x
INPUT x
ENDWHILE
OUTPUT Result
```

Mark as follows:
If no change attempted – no marks
If a change has been attempted:
Correct declarations and output statements **1 mark**
Moving INPUT x within the loop to the end of the loop
Or IF x <> –1 THEN **1 mark**
If loop works and Initialisations correct **1 mark**

**[3]**

**3 (a) (i)** Row     **2**
Column **3** **[1]**

**(ii)** NULL /  " " (empty string)/ any char other than '0', 'O' or 'X' **[1]**

**(iii)** Mark as follows:
Correct identifier
Correct dimensions
Correct data type in declaration (Correct for initial value used below)
Correct outer loop
Correct inner loop
Correct indices for assignment (LH)
Assign initial value within loop (RH) if not a valid initial value

Example Pascal

```
VAR Grid : ARRAY [1..3, 1..3] OF CHAR;
FOR Row := 1 TO 3 DO
  FOR Column := 1 TO 3 DO
    Grid[Row, Column] := NULL;
```

[7]

(b) (i) *Invalid with correct reason 1 mark each*
2,2: valid
0,1: invalid because row below range **1 mark**
1,1: valid **1 mark for the two valid cases**
1,4: column above range **1 mark**
4,1: row above range **1 mark**
2,0: Column below range **1 mark**
2,2: Cell already used (row & column within range) **1 mark**

[6]

    (ii) **Mark as follows:**
- Correct function identifier and ending
- Function parameter and return data type
- Check row within range AND
- Check column within range
- Check cell is empty (requires correct logical structure)
- Correct return values (accept TRUE/FALSE as strings)

Example Pascal
```
FUNCTION IsInputValid (Row, Column : INTEGER) : BOOLEAN;
    VAR IsValid : BOOLEAN;
    BEGIN
        IsValid := FALSE;
        IF (Row>0) AND (Row<4)
            THEN
                IF (Column>0) AND (Column<4)
                    THEN
                        IF Grid[Row, Column] = NULL  // or equivalent
                            THEN IsValid := TRUE;
        IsInputValid := IsValid
    END;
```
[max 5]

(c) (i) Use of functions/procedures **[1]**

    (ii) Eas**ier** to solve (by breaking down into sub-problems)
Can focus on one part at a time
eas**ier** to produce module code **[1]**

    (iii) Assignment: 1 / 2 / 10 / 15
Selection: 8(–20) / 13(–19)
Iteration: 5(–21) / 21
Function call: 8 / 13
Procedure call: 3 / 6 / 7 / 18 **[5]**

    (iv)
- indentation
- meaningful identifier/variable names
- keywords in capitals
- inclusion of white space
- initialising variables
- one statement per line
- use of functions/procedures with meaningful identifiers // use of structured constructs

[max 2]

(v)

Identifier	Variable or Procedure or Function or Array	Data Type	Description
GameEnd	Variable	BOOLEAN	FALSE if game in progress TRUE if there is a winner or the grid is full
Grid	ARRAY	CHAR character STRING(1)	To store the current state of the game
CurrentPlayer	Variable	CHAR character STRING(1)	The marker value ('O' or 'X')of the current player
PlayerTakesTurn	PROCEDURE	(ignore)	Current player chooses cell Program checks if it is valid and stores marker
DisplayGrid	PROCEDURE	(ignore)	Outputs the contents of the grid
HasPlayerWon	FUNCTION	BOOLEAN	Checks if the current player has completed a row, column or diagonal
GridFull	FUNCTION	BOOLEAN	Checks if the grid is full
SwapPlayer	**PROCEDURE**	(ignore)	Swaps the value of CurrentPlayer

[5]

(d) Mark as follows:
- Procedure heading and ending
- parameter given
- Byref (parameter)
- Parameter data type as CHAR (accept string)
- IF 'O' then 'X'
- IF 'X' then 'O'

```
PROCEDURE SwapPlayer (BYREF Player : CHAR)
    IF Player = 'O'
        THEN Player ← 'X'
        ELSE Player ← 'O'
    ENDIF
ENDPROCEDURE
```
[max 5]

(e) FUNCTION HasPlayerWon() RETURNS BOOLEAN accept AS or : or just BOOLE
```
    DECLARE WinningLine : BOOLEAN
    DECLARE i : INTEGER
    WinningLine ← FALSE
        // check both diagonals
    IF    Grid[1,1] = Grid[2,2] AND Grid[1,1] = Grid[3,3]
       OR Grid[1,3] = Grid[2,2]
          AND Grid[1,3] = Grid[3,1]
      THEN WinningLine ← TRUE
      ELSE
          i ← 0
        REPEAT accept LOOP / DO / / DO LOOP
          i ← i + 1
            // check a row
          IF  Grid[i,1] = Grid[i,2] AND Grid[i,1] =
            // check a column
          OR Grid[1,i] = Grid[2,i]
             AND Grid[1,i] = Grid[3,i]
            THEN WinningLine ← TRUE
          ENDIF
        UNTIL WinningLine = TRUE OR i=3
        ENDIF
      RETURN WinningLine
    ENDFUNCTION
```

1 mark

Ensure the following
elements have been
checked:
Grid[1,3]
Grid[3,1]
Grid[2,2]
Grid[3,1] =

1 mark

Ensure the following
elements have been
checked:
Grid[1,i]
Grid[2,i]
Grid[3,i]
Grid[3,1] =

[max 10]

(f) Example Pascal:

```
PROCEDURE DisplayGrid;
  BEGIN
     EmptyCell:= '' // value for empty cell see 3(a)(ii)
     FOR Row := 1 TO 3 DO
       BEGIN
         Line := '';          // build up a row for output
         FOR Column := 1 TO 3 DO
           IF Grid[Row, Column] = EmptyCell
             THEN
               Line := Line + ' : '
             ELSE
               Line := Line + ' ' + Grid[Row, Column] + ' ';
         WriteLn(Line);
       END;
  END;
```

Mark as follows:
- Procedure header & ending
- Assign empty cell value to EmptyCell
- Correctly nested loops
- Correct Boolean expression in IF statement
- Correct string concatenation $\times$ 2
- Initialise line and output line   **[max 5]**

(g) Mark as follows:
- Display of 3$\times$3 grid to represent the current state of the game
- Input box/ drop-down box for row number clearly labelled   (Accept radio buttons)
- Input box/ drop-down box for column number clearly labelled (Accept radio buttons)
- Indication of which player's turn   (Do not accept radio buttons / check boxes)
- Error message if invalid input

                                                                                      [max 4]

(h) Any **two** from:
- System testing
- Integration testing
- Black box testing
- White box testing // glass box testing   **[max 2]**