

CAMBRIDGE INTERNATIONAL EXAMINATIONS

Cambridge International Advanced Subsidiary and Advanced Level

MARK SCHEME for the May/June 2015 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 May/June 2015 series for most Cambridge IGCSE[®], Cambridge International A and AS Level components and some Cambridge O Level components.

® IGCSE is the registered trademark of Cambridge International Examinations.

Page 2	Mark Scheme	Syllabus	Paper
	Cambridge International AS/A Level – May/June 2015	9691	23

- 1 (a) (i) 'D' [1]
- (ii) Error [1]
- (iii) "FRED" [1]

(b) (i) Example solution:

```
Reverse ← ""
NumberOfLetters ← LENGTH(Original)
FOR ThisLetter ← 1 TO NumberOfLetters
    Letter ← MID(Original, ThisLetter, 1)
    Reverse ← CONCAT(Letter, Reverse)
ENDFOR
```

Marks as follows:

- Initial value of reverse is empty string
- Find length of string
- Loop for each letter
- Extract a single letter of the original string
- Build up reverse string

[max 5]

(ii) IF Original = Reverse [1]

2 (a) (i) Mark as follows:

- Line 03 1 mark
- Line 04 1 mark
- Line 07 1 mark
- Line 08 1 mark

```
01 CALL InitialiseArray() // blank board
02 CALL InputBoardDesign() // add slides and ladders data
03 TotalMoves ← 0
04 FOR Game ← 1 TO 1000
05     // play next game and update TotalMoves
06     TotalMoves ← TotalMoves + NumberOfMovesInThisGame()
07 ENDFOR // NEXT // NEXT Game
08 AverageMovesPerGame ← TotalMoves/1000
09 OUTPUT AverageMovesPerGame
```

[4]

(ii) use of procedure calls [1]

- (iii)
- easier to solve (reduce complexity) by breaking down into sub-problems
 - can focus on one part at a time
 - easier to produce module code

[max 1]

Page 3	Mark Scheme	Syllabus	Paper
	Cambridge International AS/A Level – May/June 2015	9691	23

- (iv) • *Assignment 03 / 06 / 08*
• *Iteration 04 (-07)*
• *function call 06* [3]

- (v) *TotalMoves, Game, AverageMovesPerGame*
1 mark for 1 or 2 correct variable identifiers, 2 marks for all 3 correct [2]

- (b) (i) the same number as the index
Justification: contents of array element acts as a pointer, so if no slide/ladder then position is same as index.
Alternative answer:
0 // zero // -1
Justification: if content of element is 0 then no slide/ladder, so no change of position. [2]

- (ii) Marks as follows:
• *correct index range*
• *correct data type*

Examples

Python: `Board = [0] * 31`
`Board = [0 for i in range(31)]`
Pascal: `VAR Board : ARRAY[1..30] OF INTEGER;`
Java / C#: `int[] Board = new int[30];`
C++: `int Board[30];`
VB.NET / VB6: `Dim Board(30) As Integer` [2]

- (iii) Marks as follows:
• *correct loop from 1 to 30 (accept REPEAT or WHILE loops that work)*
• *assignment of initial value to array element (allow ft from part (i))*

Example Pascal

`FOR i := 1 to 30 DO`
`Board[i] := i; // or zero or -1` [2]

Page 4	Mark Scheme	Syllabus	Paper
	Cambridge International AS/A Level – May/June 2015	9691	23

(c) Marks as follows:

- *loop (REPEAT or WHILE)*
- *Read number pairs*
- *Correct termination on input of rogue value*
- *Assign value b to Board[a]*

Example solution:

```
INPUT a
INPUT b
WHILE NOT (a = 0 AND b = 0)
    Board[a] ← b
    INPUT a
    INPUT b
ENDWHILE
```

[max 4]

(d) (i) `NumberRolled ← RANDOM(5) + 1`

[1]

(ii) Marks as follows:

- *declaration of local variables*
- *Initialisation player position*
- *initialise and update MovesSoFar*
- *Boolean expression in IF statement*
- *update player position*
- *update position if slide or ladder*
- *Boolean expression following UNTIL*
- *RETURN value*

```
FUNCTION NumberOfMovesInThisGame()
    DECLARE PlayerPosition : INTEGER
    DECLARE MovesSoFar : INTEGER
    DECLARE NumberRolled : INTEGER
    PlayerPosition ← 1
    MovesSoFar ← 0
    REPEAT
        NumberRolled ← RANDOM(5) + 1
        MovesSoFar ← MovesSoFar + 1
        // check that move does not go beyond final square
        IF PlayerPosition + NumberRolled <= 30
            THEN // make move
                PlayerPosition ← PlayerPosition + NumberRolled
                // check for slide or ladder and, if required, move
                // IF Board[PlayerPosition] > 0
                THEN
                    PlayerPosition ← Board[PlayerPosition]
                ENDF
            ENDF
        UNTIL PlayerPosition = 30
        RETURN MovesSoFar // NumberOfMovesInThisGame ← MovesSoFar
    ENDFUNCTION
```

[8]

Page 5	Mark Scheme	Syllabus	Paper
	Cambridge International AS/A Level – May/June 2015	9691	23

(e) Marks as follows:

- *Procedure heading and ending*
- *Local variable for file handle*
- *Assign file name to file handle*
- *Open file for writing*
- *Loop 1 to 30*
- *Save array elements to file*
- *Save AverageMovePerGame to file*
- *close file*

Example Pascal:

```
PROCEDURE SaveBoardDesign;
VAR FileA: TextFile;
BEGIN
    Assign (FileA, 'Design.txt');
    Rewrite(FileA);
    FOR i := 1 to 30 DO
        Writeln(FileA, Board[i]);
    Writeln(FileA, AverageMovesPerGame);
    CloseFile (FileA);
END;
```

[max 5]

(f) declare a constant maxsize

Where code requires the number of squares of the board, use this constant
 For example loop for initialising array / checking whether player has reached final square
 Only need to change value of constant if board size changes

[max 2]

3 (a) (i)

				Numbers					
i	j	Numbers[j] > Numbers[j + 1]	w	[1]	[2]	[3]	[4]	[5]	
				49	98	36	70	51	Marks:
1	1	FALSE							1
	2	TRUE	98		36	98			
	3	TRUE	98			70	98		
	4	TRUE	98				51	98	
2	1	TRUE	49	36	49				1
	2	FALSE							
	3	TRUE	70			51	70		
	4	FALSE							
3	1	FALSE							1
	2	FALSE							
	3	FALSE							
	4	FALSE							
4	1	FALSE							1
	2	FALSE							
	3	FALSE							
	4	FALSE							
1	1	1		1		1		1	Marks

Mark by row as shown. If no marks, mark by column.

[6]

- (ii) • sorts // bubble sort
• into ascending order

[2]

- (iii) 2 iterations

[1]

- (iv) • Boolean expression is evaluated repeatedly // checks array contents repeatedly
• when no more swaps are required // when the array is already sorted

[2]

Page 7	Mark Scheme	Syllabus	Paper
	Cambridge International AS/A Level – May/June 2015	9691	23

(v)

```

n ← 4
REPEAT
    NoMoreSwaps ← TRUE
    FOR j ← 1 TO n
        IF Numbers[j] > Numbers[j + 1]
            THEN
                w ← Numbers[j]
                Numbers[j] ← Numbers[j + 1]
                Numbers[j + 1] ← w
                NoMoreSwaps ← FALSE
            ENDIF
        ENDFOR
    n ← n - 1
UNTIL NoMoreSwaps = TRUE

```

Marks as follows:

- Upper bound of FOR loop set to n
- Decrement n after FOR loop
- Set Boolean variable to TRUE in outer loop, before inner loop
- Set Boolean variable to FALSE within THEN part
- UNTIL expression correct

[5]

- (b) (i)
- Indentation
 - Keywords in capitals

[max 1]

- (ii) Meaningful identifiers
Annotation/comments/remarks
Use constants (for array boundaries)

[max 1]

Page 8	Mark Scheme	Syllabus	Paper
	Cambridge International AS/A Level – May/June 2015	9691	23

4 (a) Example Pascal:

```

FUNCTION IsLeapYear(Year: INTEGER) : BOOLEAN;
BEGIN
    IF (Year MOD 400) = 0
    THEN
        IsLeapYear := TRUE
    ELSE
        IF (Year MOD 100) = 0
        THEN
            IsLeapYear := FALSE
        ELSE
            IF (Year MOD 4) = 0
            THEN
                IsLeapYear := TRUE
            ELSE
                IsLeapYear := FALSE;
        END;
    END;

```

Marks as follows:

- *function heading*
- *Correct use of MOD x 3 (Python, C uses %)*
- *Nested IFs x 3*
- *Correct RETURN values x 4 (VB assign to identifier)*
- *Indentation*

[5]

- (b)
- *A year that is divisible by 400 (TRUE)*
 - *A year that is divisible by 100, but not 400 (FALSE)*
 - *A year that is divisible by 4, but not 100 (TRUE)*
 - *A year that is not divisible by 4 (FALSE)*

[4]

Justification must match data value

- (c)
- *Integration testing*
 - *Black box testing*

[2]