
COMPUTING

9691/22

Paper 2 Written Paper

May/June 2016

MARK SCHEME

Maximum Mark: 75

Published

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 2016 series for most Cambridge IGCSE[®], Cambridge International A and AS Level components and some Cambridge O Level components.

1 [6]

String1	String2	Position	Digit1	Digit2	Sum	Carry	Result
"011101"	"001100"					"0"	" "
		6	"1"	"0"	"1"		"1"
		5	"0"	"0"	"0"		"01"
		4	"1"	"1"	"0"	"1"	"001"
		3	"1"	"1"	"1"		"1001"
		2	"1"	"0"	"0"		"01001"
		1	"0"	"0"	"1"	"0"	"101001"

1 mark for each of columns 3 to 8.

2 (a) (i) It calls itself in line 06 // Max [1]

In line 06 the function name is on the right hand side of the assignment expression

(ii) Base case: 04 / 02 (1) [2]

General case: 06 (1)

(b) (i) 1 [1]

(ii) 3 [1]

2 (c) (i) The stopping condition / base case is never reached [2]

So the function keeps calling itself for ever

(ii) IF Exponent < 0 (1) [2]

THEN

 Error (1)

ELSE ...

Or:

– check for exponent less than 0 (1)

– send error code // write function to manage negative exponents. (1)

Page 3	Mark Scheme	Syllabus	Paper
	Cambridge International AS/A Level – May/June 2016	9691	22

(d) No marks for recursive solutions [4]

```

FUNCTION Power (Number : INTEGER, Exponent : INTEGER) RETURNS INTEGER
  Result ← 1
  IF Exponent > 0
    THEN
      FOR e ← 1 to Exponent
        Result ← Result * Number
      ENDFOR
    ENDIF
  RETURN Result
ENDFUNCTION

```

Alternative:

```

FUNCTION Power (Number : INTEGER, Exponent : INTEGER) RETURNS INTEGER
  Result ← 1
  IF Exponent > 0
    THEN
      e ← Exponent
      REPEAT
        Result ← Result * Number
        e ← e - 1
      UNTIL e = 0
    ENDIF
  RETURN Result
ENDFUNCTION

```

```

FUNCTION Power (Number : INTEGER, Exponent : INTEGER) RETURNS INTEGER
  Result ← 1
  e ← Exponent
  WHILE e > 0
    Result ← Result * Number
    e ← e - 1
  ENDWHILE
  RETURN Result
ENDFUNCTION

```

(e) Iterative [2]

- iterative solution easier to write/debug
- smaller overheads (Max 1)

Recursive

- recursive solution elegant
- mathematically intuitive
- usually contains fewer lines (Max 1)

(f) (i) - in the main program just before the function is called (1) [2]

- to then single-step the function code (1)

Page 4	Mark Scheme	Syllabus	Paper
	Cambridge International AS/A Level – May/June 2016	9691	22

- (ii) – Result – this is the value to be returned at the end of each call (1) [2]
 – Exponent – has a different value each time the function is called (1)

alternative marking:

- Result, Exponent
- these variables change in the program

- (iii) – from the breakpoint / set one breakpoint [3]
 – step one instruction at a time
 – inspecting the variable watch after each instruction

- 3 (a) (i) White: 0 [2]
 Black: –1 / NULL
 Accept any other appropriate integer value (e.g. White –1, Black 0)

- (ii) **Example VB:** [3]

```
DIM Puzzle(11,11) AS INTEGER
```

Example Python:

```
Puzzle = [[0 for i in range(12)] for j in range(12)]
Puzzle = [[0]*11]*11
```

Example Pascal:

```
VAR Puzzle : Array[1..11, 1..11] OF INTEGER;
```

Example C and C++:

```
int Puzzle[11][11];
```

Example C#:

```
int [11][11] Puzzle;
```

Mark as follows:

- correct identifier
- correct dimensions
- integer data type

Page 5	Mark Scheme	Syllabus	Paper
	Cambridge International AS/A Level – May/June 2016	9691	22

(iii) Example VB:

[3]

```
FOR i = 1 TO 11
  FOR j = 1 TO 11
    Puzzle(i,j) = 0
  NEXT j
NEXT i
```

Example Python:

```
Puzzle = [[0 for i in range(12)] for j in range(12)]
```

Example Pascal:

```
FOR i := 1 TO 11 DO
  FOR j := 1 TO 11 DO
    Puzzle[i,j] := 0 ;
```

Example C++:

```
for (int i = 1; i <= 11; i++)
  for (int j = 1; j <=11; j++){
    Puzzle[i][j] = 0;}
```

Mark as follows:

- looping 11 times
- correctly nested inner loop
- correct assignment of array element with value for White (f.t.)

(iv) Example VB:

[2]

```
Puzzle(1,7) = -1
```

Example Python:

```
Puzzle[1][7] = -1
```

Example Pascal:

```
Puzzle[1,7] := -1;
```

Example C++:

```
Puzzle[1][7] = -1;
```

Mark as follows:

- identifier with indexes
- assignment of value for black

(b) (i) CONSTANT WHITE = 0 // value from part(a)(i) (1) [2]

CONSTANT BLACK = -1 // value from part(a)(i) (1)

(ii)

Max [8]

```

PROCEDURE CheckForStartOfWord(Puzzle, ThisRow, ThisColumn,
                               Across, Down)
  Across ← FALSE // will change to TRUE
                // if a word across starts in this square
  Down ← FALSE                                     (1)
  IF Puzzle[ThisRow, ThisColumn] = WHITE         (1)
    THEN // this square is white
      // check for sequence across
      IF ThisColumn < 11 // check not in last column
        THEN
          // check this is the first column or a black square to the left
          IF (ThisColumn = 1
              OR Puzzle[ThisRow, ThisColumn - 1] = BLACK)
            AND (Puzzle[ThisRow, ThisColumn + 1] = WHITE)   (1)
            THEN
              Across ← TRUE
            ENDIF                                     (1)
          ENDIF
        ENDIF
      // check for sequence down
      IF ThisRow < 11 // check not in last row
        THEN
          // check this is the first row or a black square above
          IF (ThisRow = 1                                     (1)
              OR Puzzle[ThisRow - 1, ThisColumn] = BLACK)   (1)
            // check that the square below is white
            AND (Puzzle[ThisRow + 1, ThisColumn] = WHITE)   (1)
            THEN
              Down ← TRUE                                     (1)
            ENDIF
          ENDIF
        ENDIF
      ENDIF
    ENDIF
  ENDPROCEDURE

```

(iii)

[3]

Parameter	By reference	By value	
Puzzle	✓		(1)
ThisRow		✓	(1)
ThisColumn		✓	(1)
Across	✓		(1)
Down	✓		(1)

Page 7	Mark Scheme	Syllabus	Paper
	Cambridge International AS/A Level – May/June 2016	9691	22

(c) Example VB:

[8]

```

NextNumber = 1
a = 1
d = 1
FOR ThisRow = 1 TO 11      FOR ThisColumn = 1 TO 11
    CheckStartOfWord(ThisRow, ThisColumn, Across, Down)
    IF Across = TRUE THEN
        AcrossList(a) = NextNumber
        a = a + 1
    END IF
    IF Down = TRUE THEN
        DownList(d) = NextNumber
        d = d + 1
    END IF
    IF (Across = TRUE) OR (Down = TRUE) THEN
        Puzzle(ThisRow, ThisColumn) = NextNumber
        NextNumber = NextNumber + 1
    END IF
NEXT ThisColumn
NEXT ThisRow

```

Example Python:

```

NextNumber = 1
a = 1
d = 1
for ThisRow in range(1, 12):
    for ThisColumn in range(1,12):
        CheckStartOfWord(ThisRow, ThisColumn, Across, Down)
        if Across:
            AcrossList[a] = NextNumber
            a = a + 1
        if Down == True:
            DownList[d] = NextNumber
            d = d + 1
        if (Across == True) or (Down == True):
            Puzzle[ThisRow][ThisColumn] = NextNumber
            NextNumber = NextNumber + 1

```

Page 8	Mark Scheme	Syllabus	Paper
	Cambridge International AS/A Level – May/June 2016	9691	22

Example Pascal: (1)

```

NextNumber := 1;
a := 1;
d := 1;
FOR ThisRow := 1 TO 11 DO
  FOR ThisColumn := 1 TO 11 DO
    BEGIN
      CheckStartOfWord(ThisRow, ThisColumn, Across, Down);
      IF Across = TRUE
        THEN
          BEGIN
            AcrossList[a] := NextNumber;
            a := a + 1;
          END;
      IF Down = TRUE
        THEN
          BEGIN
           DownList[d] := NextNumber;
            d := d + 1;
          END;
      IF (Across = TRUE) OR (Down = TRUE)
        THEN
          BEGIN
            Puzzle[ThisRow, ThisColumn] := NextNumber;
            NextNumber := NextNumber + 1;
          END;
    END;
  END;
END;

```

Mark as follows:

- all 3 initialisations
- outer loop correctly formed
- inner loop correctly nested
- procedure call with all parameters
- 3 IF statements, not nested
- assign NextNumber to AcrossList and DownList
- increment a, d, NextNumber
- assign NextNumber to Puzzle element

Page 9	Mark Scheme	Syllabus	Paper
	Cambridge International AS/A Level – May/June 2016	9691	22

- (d) – constant declaration Max [4]
– meaningful identifiers/variable names
– modules // procedure calls
– use of parameters
– indentation
– capitalised variable names/identifiers
– upper case keywords // capitalisation of keywords
– annotation

(e) Example VB.NET: Max [5]

```

SUB SavePuzzleToFile(Puzzle) (1)
    DIM FileWriter AS StreamWriter (1)
    DIM Row, Column AS INTEGER (1)
    FileWriter = New StreamWriter("Puzzle.TXT") (1)
    FOR Row = 1 TO 11
        FOR Column = 1 TO 11
            FileWriter.Write(Puzzle(Row, Column)) (1)
        NEXT Column
        FileWriter.WriteLine()
    NEXT Row (1)
    FileWriter.Close() (1)
END SUB

```

Example VB6:

```

Sub SavePuzzleToFile(Puzzle)
    Dim i as Integer
    Open "Puzzle.TXT" For Output As #1
    For i = 1 To 11
        For j = 1 TO 11
            Write #1, Puzzle(i,j)
        Next j
    Next i
    Close#1
End Sub

```

Example Python:

```

def SavePuzzleToFile(Puzzle) :
    PuzzleFile = open("Puzzle.TXT", "w")
    for i in range(1,12) :
        for j in range(1,12):
            PuzzleFile.write(str(Puzzle[i][j]))
        PuzzleFile.write("\n")
    PuzzleFile.close()

```

Page 10	Mark Scheme	Syllabus	Paper
	Cambridge International AS/A Level – May/June 2016	9691	22

Example Pascal:

```

PROCEDURE SavePuzzleToFile (Puzzle);           (1)
BEGIN
  VAR PuzzleFile : TEXTFILE;
  VAR Row, Column : INTEGER;                 (1)
  ASSIGNFILE (PuzzleFile, 'Puzzle.TXT');     (1)
  REWRITE (PuzzleFile);                     (1)
  FOR Row := 1 TO 11 DO
    FOR Column := 1 TO 11 DO                (1)
      WRITE(PuzzleFile, Puzzle[Row, Column]); (1)
    CLOSEFILE (PuzzleFile);                 (1)
  END;

```

Mark as follows:

- procedure heading and ending
- declaration of local variables
- assigning a file name
- open file for writing
- nested loop to access each array element
- write element out to file
- close file

(f)

Max [7]

```

FUNCTION CountSquaresAcross (Puzzle, ThisRow, ThisColumn) RETURNS
INTEGER

  DECLARE WordLength : INTEGER
  WordLength ← 2 // this was the minimum word length
  WHILE Puzzle[ThisRow, ThisColumn + WordLength] = WHITE
    AND (ThisColumn + WordLength) <= 11
    WordLength ← WordLength + 1
  ENDWHILE
  RETURN WordLength
ENDFUNCTION

```

Mark as follows:

- declaration of local variable
- initialise counter
- loop using WHILE or REPEAT
- increment counter
- check for white square
- check for right edge of puzzle
- return counter
- end function