CAMBRIDGE INTERNATIONAL EXAMINATIONS

GCE Advanced Subsidiary Level and GCE Advanced Level

MARK SCHEME for the May/June 2014 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 2014 series for most IGCSE, GCE Advanced Level and Advanced Subsidiary Level components and some Ordinary Level components.

CAMBRIDGE
International Examinations

1 (a) (i) *Mark as follows:*

1 mark for heading/introduction
1 mark for method of displaying numbers 1 – 10
e.g. dropdown box/radio buttons/grid of numbers)
1 mark for method to move to next screen (ignore exit/cancel) (accept next/ok/enter)
e.g. button // a label telling child what to do [3]

(ii) 1 mark for explanation that fits design of (a)(i) [1]
e.g. clicking on/touching/pressing button/box/number/icon

(b) Mark as follows:

- 1 mark for each box correctly translated into chosen programming language
- Identifiers must be the same as given in flowchart
- Give 1 mark for loop header and end correctly coded (must be a FOR loop)
- Ignore any declarations
- If candidate only says “Visual Basic” (no version number) use the mark scheme that best fits the answer
- If language given is “Pseudocode”, give no marks

[max 6]

VB6 – accept console mode answers

```
Number = InputBox("")
Msg = Number & " Times Table" & vbCrLf
For i = 1 To 10
   Result = i * Number
   Msg = Msg & i & "x" & Number & "=" & Result & vbCrLf
Next i
Msg = Msg & "Press any key"
MsgBox(Msg)
```

VB.NET/VB 2005 *etc.*

```
Number = Console.ReadLine();
Console.WriteLine(Number & " Times Table");
For i = 1 To 10
Result = i * Number
Console.WriteLine(i & "x" & Number & "=" & Result)
Next i
Console.WriteLine("Press any key")
```

QBASIC

```
INPUT Number;
PRINT Number; " Times Table"
FOR i = 1 TO 10
   Result = i * Number
   PRINT i; "x"; Number; "="; Result
NEXT i
PRINT "Press any key"
```

PASCAL

```
ReadLn(Number);
WriteLn(Number, ' Times Table');
For i := 1 To 10 Do
Begin
   Result := i * Number;
   WriteLn(i, 'x', Number, '=', Result);  // semicolon optional here
End;                                  // Begin End are part of the loop
structure
WriteLn('Press any key');
```

PYTHON

```
Number = int(input())
print(Number, "Times Table")
for i in range(1,11) :
    Result = i * Number
    print(i, "x", Number, "=", Result)
print("Press any key")
```

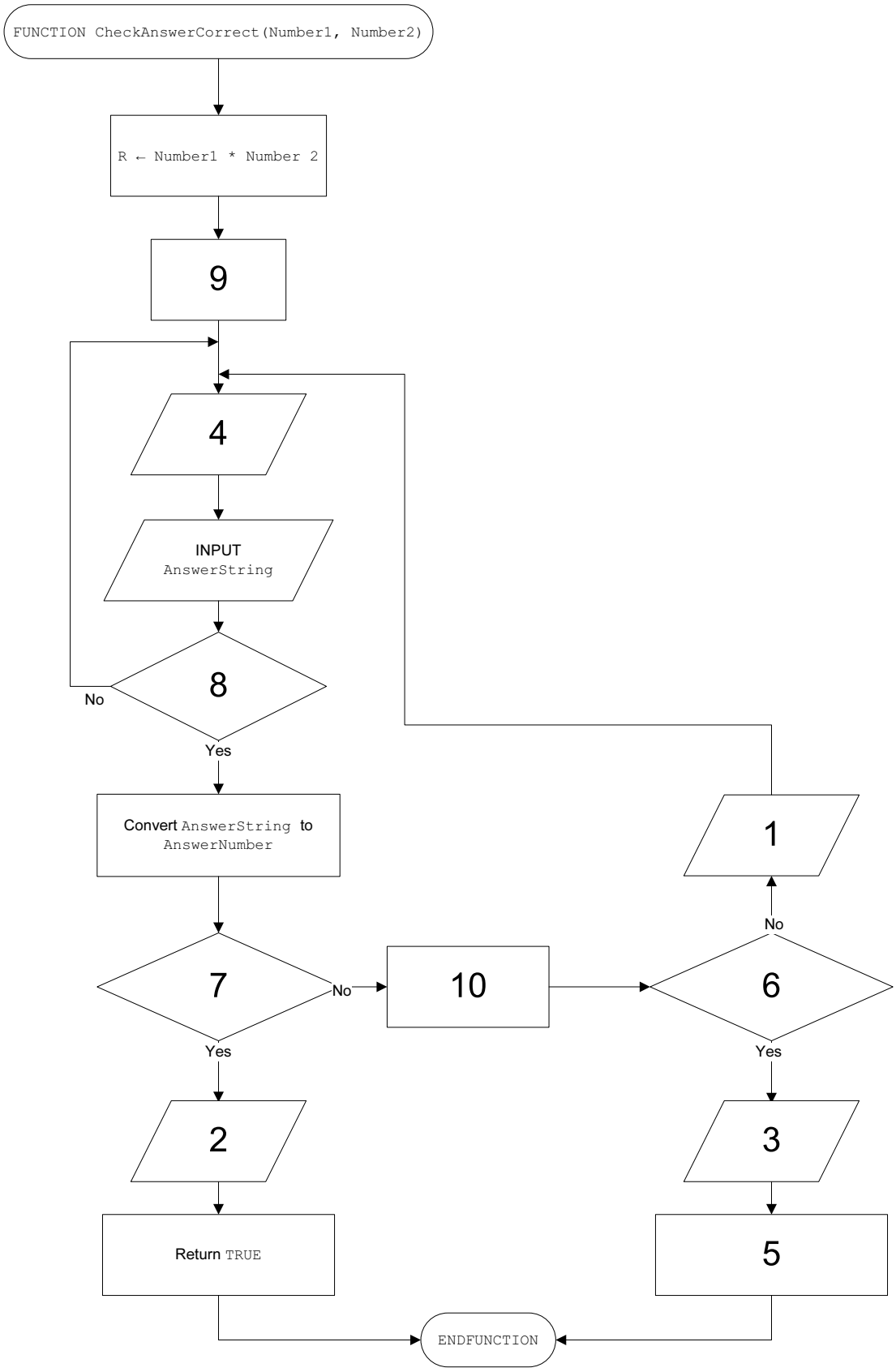
(c)

```
PROCEDURE ShowMultiplicationGrid (Number1, Number2)
   FOR Row ← 1 TO Number1              // 1 mark
      FOR Column ← 1 TO Number2        // 1 mark
         OUTPUT '*'      // 1 mark – accept '*' or "*"
      ENDFOR                          // 1 mark – accept NEXT column
      OUTPUT NewLine                  // 1 mark – must be in outer loop
   ENDFOR
ENDPROCEDURE
```

1 mark if `Number1` and `Number2` switched round

[5]

(d) 1 mark for each correctly labelled shape (accept full text instead of number)



[10]

(2) (a)

```
FUNCTION TestScoreTotal RETURNS INTEGER          // 1 mark
  DECLARE AnswerCorrect, Finish : BOOLEAN        // 1 mark
  DECLARE Number1, Number2, Score : INTEGER  // 1 mark
  Score ← 0                                      // 1 mark

  Finish ← FALSE
  REPEAT
    Number1 ← Random(10)      // accept X instead of 10
    Number2 ← Random(10)    // 1 mark
    Display(Number1, Number2)
    AnswerCorrect ← CheckAnswerCorrect(Number1, Number2)
    IF AnswerCorrect = TRUE                    // 1 mark
      THEN Score ← Score + 1
      ELSE Finish ← TRUE                       // 1 mark
    ENDIF                                                  // accept =
instead of ←
  UNTIL Finish = TRUE                          // 1 mark
  RETURN Score                                   // 1 mark
ENDFUNCTION
```

[9]

- If candidate only says “Visual Basic” (no version number) use the mark scheme that best fits the answer
- If language given is “Pseudocode”, give no marks
- Identifiers must be as given in the question

Mark as follows:

(b) (i) 2 marks
- Correct dimension 1 mark
- Correct data types 1 mark [2]

(ii) 2 marks
- 1 mark for structure
- 1 mark for correct fields (`Name, BestScore`) listed [2]

(iii) 2 marks
- 1 mark for correct name (`Student`) and size
- 1 mark for correct record type (`StudentScore`) [2]

(iv) 3 marks
- 1 mark for correctly addressed array elements (`Student(3)`)
- 1 mark for correct reference to individual fields (`.Name, .BestScore`)
- 1 mark for correct assignment of values [3]

VB6

QUESTION 2(b)(i)

```
Dim Name(30) As String
Dim BestScore(30) As Integer
```

QUESTION 2(b)(ii)

```
Type StudentScore
       Name As String
       BestScore As Integer
End Type
```

QUESTION 2(b)(iii)

```
Dim Student(30) As StudentScore
```

QUESTION 2(b)(iv)

```
Student(3).Name = "Anji"
Student(3).BestScore = 15
```

VB.NET/VB 2005 *etc.*

QUESTION 2(b)(i)

```
Dim Name(30) As String
Dim BestScore(30) As Integer
```

QUESTION 2(b)(ii)

```
Structure StudentScore
       Dim Name As String
       Dim BestScore As Integer
End Structure
```

QUESTION 2(b)(iii)

```
Dim Student(30) As StudentScore
```

QUESTION 2(b)(iv)

```
Student(3).Name = "Anji"
Student(3).BestScore = 15
```

QBASIC

QUESTION 2(b)(i)

```
DIM Name(30) AS STRING * 15
DIM BestScore(30) AS INTEGER
```

QUESTION 2(b)(ii)

```
TYPE StudentScore
       Name AS STRING * 15
       BestScore AS INTEGER
END TYPE
```

QUESTION 2(b)(iii)

```
DIM Student(30) AS StudentScore
```

QUESTION 2(b)(iv)

```
Student(3).Name = "Anji"
Student(3).BestScore = 15
```

PASCAL

QUESTION 2(b)(i)

```
Name : Array[1..30] Of String;
BestScore : Array[1..30] Of Integer;
```

QUESTION 2(b)(ii)

```
Type StudentScore = Record
            Name : String[15];
            BestScore : Integer;
     End;
```

QUESTION 2(b)(iii)

```
Student : Array[1..30] Of StudentScore;
```

QUESTION 2(b)(iv)

```
Student[3].Name := 'Anji';
Student[3].BestScore := 15;
```

PYTHON

QUESTION 2(b)(i)

```
Name = ["" for i in range(30)]
BestScore = [0 for i in range(30)]
```

or

```
Name = []
BestScore = []
for i in range(30) :
      Name.append("")
      BestScore.append(0)
```

QUESTION 2(b)(ii)

```
class StudentScore :
    def __init__(self) :
       Name = ""
       BestScore = 0
```

QUESTION 2(b)(iii)

```
Student = [StudentScore() for i in range(30)]
```

or

```
Student = []
for i in range(30) :
   Student.append(StudentScore())
```

QUESTION 2(b)(iv)

```
Student[2].Name = "Anji"
Student[2].BestScore = 15
```

Mark as follows:

- Procedure/sub SaveToFile
- Open (for writing) StudentFile
- Initialise index value correctly
- Loop through student array
- Write record to file StudentFile
- Close file StudentFile

(c) **VB6**

```
Sub SaveToFile()
    Dim i as Integer
    Open "StudentFile" For Output As #1
    For i = 1 To 30
        Write#1, Student(i).Name, Student(i).BestScore
    Next i
    Close#1
End Sub
```

VB.NET/VB 2005 *etc.*

```
Sub SaveToFile()
    Dim Writer As BinaryWriter
    Writer     =     new     BinaryWriter(File.Open("StudentFile",
                                                FileMode.Create))
    For i = 1 To 30
        Writer.Write(Student(i).Name)
        Writer.Write(Student(i).BestScore)
    Next i
    Writer.Close()
End Sub
```

QBASIC

```
SUB SaveToFile(Student() AS StudentScore)
    OPEN "StudentFile" FOR OUTPUT AS #1
    FOR i = 1 TO 30
        WRITE #1, Student(i).Name, Student(i).BestScore
    NEXT i
    CLOSE #1
END SUB
```

PASCAL

```
Var Students : File Of StudentScore;
Procedure SaveToFile;
Var i : Integer;
Begin
    Assign(Students, 'StudentFile');
    Rewrite(Students);
    For i := 1 To 30 Do
        Write(Students, Student[i]);
    Close(Students);
End;
```

PYTHON

```
import pickle
def SaveToFile() :
    Students = open("StudentFile", "wb")
    for i in range(30) :
        pickle.dump(Student[i], Students)
    Students.close()
```

or *(as a text file)*

```
def SaveToFile() :
    Students = open("StudentFile", "w")
    for i in range(30) :
        Students.write(Student[i].Name + "\n")
        Students.write(str(Student[i].BestScore) + "\n")
    Students.close()
```

[6]

3 (a) (i)

```
FUNCTION FindArrayIndex(ChildName : STRING) RETURNS INTEGER
i ← 0
REPEAT
   i ← i + 1
UNTIL Student[i].Name = ChildName
RETURN i
```

Another method:

```
FUNCTION FindArrayIndex(ChildName : STRING) RETURNS INTEGER
   Found ← FALSE
   i ← 1
   WHILE NOT (Found = TRUE)
      IF Student[i].Name = ChildName
         THEN Found ← TRUE
         ELSE i ← i + 1
      ENDIF
   ENDWHILE
RETURN i
```

[max 5]

(ii) *suggestion of checking for end of array*
// checking whole array and setting flag if not found
Special value/error code/number returned if name doesn't exist (e.g. –1)

[2]

(b) *e.g.*
– turn all characters into lower/upper case
– before saving/searching // on input

[2]

(4) (i) – it calls itself [1]

(ii)

Call Number	Function call	s	x	RIGHT (s,x - 1)	LEFT(s,1)	Return value
1	Y('BYTE')	'BYTE'	4	'YTE'		
2	Y('YTE')	'YTE'	3	'TE'		
3	Y('TE')	'TE'	2	'E'		
4	Y('E')	'E'	1			'E'
(3)					'T'	'ET'
(2)					'Y'	'ETY'
(1)					'B'	'ETYB'

1 mark per correct column [7]

(iii) reverses the string [1]

(iv) – indentation
comment/annotation/remarks [2]

(v) – identifiers not meaningful/sensible // identifiers are just single characters [1]

(vi) if answer is recursive, no marks

There are many different ways of solving this. The following are examples:

```
n ← "/empty string
REPEAT
  x ← LENGTH(s)
  z ← LEFT(s,1)
  s ← RIGHT(s, x - 1)
  n ← z + n          // y ← concat(z,n)
UNTIL s = "
RETURN n
```

Or

```
n ← "/empty string
x ← LENGTH(s)
FOR i ← 1 TO x
  z ← LEFT(s,1)
  s ← RIGHT(s, x - i)
  n ← z + n          // n ← concat(z,n)
ENDFOR
RETURN n
```

```
n ← "/empty string
x ← LENGTH(s)
FOR i ← 1 TO x
  z ← MID(s,i,1)
  n ← z + n          // n ← concat(z,n)
ENDFOR
RETURN n
```

Mark as follows:
– start with empty string
– correct loop structure
– correct loop count/termination
– pick single character (from string s) consecutively
– concatenate single character to correct end of new string
– return newly formed string

[5]