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 2016 series for most Cambridge IGCSE®, Cambridge International A and AS Level components and some Cambridge O Level components.
1 (a)

[Flowchart diagram]

- Start
- INPUT PlayerName
- ReadPlayerTotal
- INPUT PlayerGameGrade
- CASE PlayerGameGrade
  - 'B': PointsTotal ← PointsTotal + 1
  - 'C': PointsTotal ← PointsTotal + 3
  - 'D': PointsTotal ← PointsTotal + 5
- OUTPUT PointsTotal
- SavePlayerTotal
- IS PointsTotal > 11?
  - Yes: OUTPUT "ELIMINATED"
  - No: Stop
Mark as follows:

- One mark per shape, correctly labelled (except for three assignments as noted above)
- One mark for three selection values ('B', 'C' and 'D') [9]

(b)

<table>
<thead>
<tr>
<th>Points</th>
<th>Total</th>
<th>PlayerGameGrade</th>
<th>Updated</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>A</td>
<td>n</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>B</td>
<td>n + 1</td>
<td>n + 1</td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>C</td>
<td>n + 3</td>
<td>n + 3</td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>D</td>
<td>n + 5</td>
<td>n + 5</td>
<td></td>
</tr>
<tr>
<td>e.g. 10</td>
<td>e.g. C</td>
<td>13</td>
<td>13</td>
<td>ELIMINATED</td>
</tr>
</tbody>
</table>

One mark per complete row testing different routes through the algorithm. [5]

(c)

```python
INPUT PlayerGameGrade
WHILE NOT(PlayerGameGrade = 'A' OR PlayerGameGrade = 'B' OR PlayerGameGrade = 'C' OR PlayerGameGrade = 'D')
OUTPUT "Invalid - Re-enter"
INPUT PlayerGameGrade
ENDWHILE
```

One mark for each of:

- WHILE ... ENDWHILE
- Correct condition in a loop
- INPUT within loop plus one before loop // alternative arrangement leading to correct exit from loop [3]

2 (a) (i) 'e' [1]
(ii) "Cat-food" [1]
(iii) 213 [1]

(b) (i) 03 // 3 [1]
(ii) 29 [1]
(iii) 14 // 16 [1]
(iv) 18 // 24 // 25 [1]
(c) NextChar <> '*'

(d) (i)

<table>
<thead>
<tr>
<th>i</th>
<th>j</th>
<th>NextChar</th>
<th>NextNumberString</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>'2'</td>
<td>&quot;&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>'3'</td>
<td>&quot;23&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>'*'</td>
<td></td>
<td></td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>'7'</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>'3'</td>
<td>&quot;73&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>'1'</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>'*'</td>
<td>&quot;731&quot;</td>
<td></td>
<td></td>
<td>731</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>'5'</td>
<td>&quot;&quot;</td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>'*'</td>
<td>&quot;5&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>'#'</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

One mark for each of columns 1 to 4
One mark for numbers 2 & 3 as shown in box

(ii) One mark for each of:

- Isolates / separates / splits up each numeric string / the numbers / data string separated by "*"
- Converts each numeric string / each number into an integer and
- Stores each integer in array (Numbers)
3 (a) (i) Declaration of a variable // identifier \[1\]

(ii) $\text{TimesTable, // $\text{UpTo // $\text{Posn // $\text{Product}}}$ [1]

(iii) 15 // 16 // 18 // 21 // 23 [1]

(iv) Statements inside the loop are enclosed by curly brackets {} // or by example, such as \(<\text{statements}>\) [1]

(b) (i) • a learned / existing skill…
• … which can be applied to / used in a new situation / role [2]

(ii) The ability to recognise:
• Similar syntax
  – Assignment / variables / data types
  – Common operators / symbols for functions (+, −, /, *, OR, AND, >, <…)

• Control Structures
  – Iteration
  – Selection
  – Sequence
  – Layout / format (e.g. indentation)

• Modular features
  – Objects
  – Procedures / Functions

Any two of the above. [Max. 2]
4 (a) \[ \text{INT}(\text{RND()} \times 150) + 1 \]

One mark for each part as follows:
- \( \text{RND()} \times 150 \)
- + 1
- \( \text{INT()} \)  

(b) ‘Pseudocode’ solution included here for development and clarification of mark scheme. Programming language example solutions appear in the Appendix.

**Expected Loop-based solution:**

DECLARE \( i \), NextNumber : INTEGER
FOR \( i \leftarrow 1 \) TO 4
    NextNumber \leftarrow 1 + \text{INT}(\text{Rnd()} \times 150)
    OUTPUT NextNumber
ENDFOR

Mark as follows:
- Declaration of all variables used including data types
- Loop
- Assignment / calculation of (four) different random numbers (0 to 150) in a loop
- Output of four values

**ALTERNATIVE Non-Loop version**

DECLARE Num1, Num2, Num3, Num4 : INTEGER

\[ \begin{align*}
    \text{Num1} & \leftarrow \text{INT}(\text{RND()} \times 150) + 1 \\
    \text{Num2} & \leftarrow \text{INT}(\text{RND()} \times 150) + 1 \\
    \text{Num3} & \leftarrow \text{INT}(\text{RND()} \times 150) + 1 \\
    \text{Num4} & \leftarrow \text{INT}(\text{RND()} \times 150) + 1
\end{align*} \]

OUTPUT Num1, Num2, Num3, Num4

Mark as follows:
- Declaration of all variables used including data types
- Assignment of four different random numbers (0 to 150)
- Assignment to four separate variables
- Output of four values
(c) Visual Basic

Function GenerateNumber(ByVal AnyName AS INTEGER) AS INTEGER

Pascal

FUNCTION GenerateNumber (AnyName : INTEGER) : INTEGER

Python

def GenerateNumber (AnyName):

• Mark as follows:
• Correct keyword + Function name
• Single input parameter of correct type
• Return parameter type [3]

(d) (i) • Program code is modified
• following a change to the requirements [2]

(ii) • Use an array / list / file to store each number generated // a flag value
• Check the array / list / file to see if the new random number has already been drawn
• If YES, generate another number
• If NO, output the number and update the array / list / file [Max. 3]

5 (a) • 2D array
• of type integer
• with identifier PlayerScore [Max. 2]

(b) (i) Stepwise refinement // Top-Down Design [1]


DECLARE ThisPlayerName : STRING
DECLARE PlayerName : ARRAY[1:8] OF STRING
DECLARE i : INTEGER

OPENFILE "NAMES.TXT" FOR READ
i ← 1
WHILE NOT EOF("NAMES.TXT")
    READFILE "NAMES.TXT", ThisPlayerName
    PlayerName[i] ← ThisPlayerName
    i ← i + 1
ENDWHILE

CLOSEFILE "NAMES.TXT"

One mark for each of:
• File open in read mode
• Loop until EOF() or count-controlled (8 iterations)
• Read a line from the file in a loop
• Assignment to PlayerName[1 to 8] from the file in a loop
Close file [Max. 4]

```plaintext
// search for player name ....
Found ← FALSE
i ← 1

REPEAT
    IF ThisPlayerName = PlayerName[i]
        THEN
            Found ← TRUE
            PlayerNumber ← i
        ELSE
            i ← i + 1
    ENDIF
UNTIL (Found = TRUE) OR (i = 9)
```

One mark for each of:
- Initialise i to 1 and Found to FALSE
- Loop through array PlayerName (including exit when found)
- Comparison: ThisPlayerName = PlayerName[i] in a loop
- Found set to TRUE if ThisPlayerName found

(c) (i) • a nested // an inner and an outer
• count controlled // incremented loop(s)

(ii) "...
<table>
<thead>
<tr>
<th></th>
<th>True</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>False</td>
</tr>
</tbody>
</table>

Both answers must be correct

(iii) Error line number 5, 9 or 11 as follows:

Line 5:
The boundary value must be included //
IF PlayerScore[GameIndex, PlayerIndex] >= 100 // > 99

Line 9:
The boundary value must be included //
IF PlayerScore[GameIndex, PlayerIndex] >= 50 // > 49

Line 11:
One should be added to Total50 (not GameIndex) //
Total50 ← Total50 + 1

One mark for line number + explanation
6  (i)  10 / 10.0  [1]
   (ii)  18.4  [1]
   (iii)  41  [1]
   (iv)  TRUE  [1]
   (v)  12.4  [1]
Appendix – Program code example solutions

Q4 (b): Visual Basic

Randomize()
Dim i As Integer
Dim NextNumber As Integer
For i = 1 To 4
    NextNumber = 1 + Int(Rnd() * 150)
    Console.WriteLine(NextNumber)
Next

OR

Randomize()
Dim Num1, Num2, Num3, Num4 As Integer
Num1 = 1 + Int(Rnd() * 150)
Num2 = 1 + Int(Rnd() * 150)
Num3 = 1 + Int(Rnd() * 150)
Num4 = 1 + Int(Rnd() * 150)
Console.WriteLine(Num1, Num2, Num3, Num4)

Q4 (b): Pascal

Var i : Integer;
    NextNumber : Integer;
Begin
    Randomize;
    For i := 1 To 4 Do
        Begin
            NextNumber := 1 + Random(150);
            Writeln(NextNumber);
        End;
    Readln;
End.

OR

Var Num1, Num2, Num3, Num4 : Integer;
Begin
    Randomize;
    Num1 := 1 + Random(150);
    Num2 := 1 + Random(150);
    Num3 := 1 + Random(150);
    Num4 := 1 + Random(150);
    Writeln(Num1, Num2, Num3, Num4);
    Readln;
End.
Q4 (b): Python

```python
import random
# i : Integer
# NextNumber : Integer

for i in range(1, 5) :
    NextNumber = 1 + int(150 * random.random())
    print(NextNumber)

Alternative:

import random
# i Integer
# NextNumber Integer
for i in range(1, 5) :
    NextNumber = random.randint(1, 150)
    print(NextNumber)

OR

import random
# i Integer
# Num1, Num2, Num3, Num4 Integer

Num1 = random.randint(1, 150)
Num2 = random.randint(1, 150)
Num3 = random.randint(1, 150)
Num4 = random.randint(1, 150)

print(Num1, Num2, Num3, Num4)
```
Q5 (b) (ii): Visual Basic

Dim PlayerName(8) As String
Dim i As Integer
FileOpen(1, "Names.txt", OpenMode.Input)
i = 1
Do
    PlayerName(i) = LineInput(1)
i = i + 1
Loop Until EOF(1)
FileClose(1)

Alternative:

Dim PlayerName(8) As String
Dim i As Integer
FileOpen(1, "Names.txt", OpenMode.Input)
For i = 1 To 8
    PlayerName(i) = LineInput(1)
Next
FileClose(1)

Alternative:

Dim sr As StreamReader = New StreamReader("Names.txt")
Dim line As String
line = sr.ReadLine()
i = 1
Do While (line <> Nothing)
    PlayerName(i) = line
    i = i + 1
    line = sr.ReadLine()
Loop
sr.Close()
Q5 (b) (ii): Pascal

Var Names : TextFile;
i : Integer;
PlayerName : Array[1..8] Of String;
Begin
  AssignFile(Names, 'Names.txt');
  Reset(Names);
i := 1;
While Not Eof(Names) Do
  Begin
    Readln(Names, PlayerName[i]);
    Writeln(PlayerName[i]);
i := i + 1;
  End;
End;
Close(Names);
Readln;
End.

Alternative:

Var Names : TextFile;
i : Integer;
PlayerName : Array[1..8] Of String;
Begin
  AssignFile(Names, 'Names.txt');
  Reset(Names);
  For i := 1 To 8 Do
    Begin
      Readln(Names, PlayerName[i]);
      Writeln(PlayerName[i]);
i := i + 1;
    End;
End;
Close(Names);
Readln;
End.
Q5 (b) (ii): Python

# PlayerName : List
# NextPlayer : String
# File : File handle
File = open("Names.txt", "r")
PlayerName = []
while (1) :
    NextPlayer = File.readline()
    if not NextPlayer :
        break
    else :
        PlayerName.append(NextPlayer)
File.close()

Alternative:

# PlayerName : List
# NextPlayer : String
# File : File handle
# i : Integer
File = open("Names.txt", "r")
PlayerName = []
for i in range(1, 9) :
    NextPlayer = File.readline()
    PlayerName.append(NextPlayer)
File.close()

Alternative:

# PlayerName : List
# NextPlayer : String
# File : File handle
# i : Integer
File = open("Names.txt", "r")
PlayerName = ["" for i in range(8)]
for i in range(1, 9) :
    PlayerName[i - 1] = File.readline()
File.close()
Q5 (b) (iii): Visual Basic

Found = False
i = 1

Do
    If ThisPlayerName = PlayerName(i) Then
        Found = True
        PlayerNumber = i
    Else
        i = i + 1
    End If
Loop Until Found = True Or i = 9

Q5 (b) (iii): Pascal

Begin
Found := False;
i := 1;
Repeat
    If (ThisPlayerName = PlayerName[i]) Then
        Begin
            Found := True;
            PlayerNumber := i;
        End
    Else
        i := i + 1;
    Until (Found) Or (i = 9);
End.

Q5 (b) (iii): Python

Found = FALSE
PlayerName = [j.strip() for j in PlayerName]
if ThisPlayerName in PlayerName :
    PlayerNumber = PlayerName.index(ThisPlayerName) + 1
    Found = TRUE

Alternative:

Found = False
i = 1
while not Found and i < 9 :
    if ThisPlayerName == PlayerName[i].strip() :
        Found = True
        PlayerNumber = i
    else :
        i = i + 1

Alternative:

Found = False
for i in range(1, 9) :
    if ThisPlayerName == PlayerName[i].strip() :
        Found = True
        PlayerNumber = i