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 2015 series for most Cambridge IGCSE®, Cambridge International A and AS Level components and some Cambridge O Level components.
1. (i) 40  
(ii) 314.2(0)  
(iii) 16 // ERROR as identifier Z has not been declared  
(iv) TRUE

2. (i) (Single) software program  
Features for:  
- program editor/writing/editing  
- translation // interpreter/compiler  
- testing program code // observe outputs  
2 points to score  

(ii) Syntax checking (on entry)  
Structure blocks (e.g. IF structure and loops begin/end highlighted)  
General prettyprint features  
Automatic indentation  
Highlights any undeclared variables  
Highlights any unassigned variables  
Commenting out/in of blocks of code  
Visual collapsing / highlighting of blocks of code  
Single stepping  
Breakpoints  
Variable/expressions report window [MAX 3]
3 (a)  

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>InA</td>
<td>InB</td>
</tr>
<tr>
<td>1</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
<tr>
<td>2</td>
<td>TRUE</td>
<td>FALSE</td>
</tr>
<tr>
<td>3</td>
<td>FALSE</td>
<td>TRUE</td>
</tr>
<tr>
<td>4</td>
<td>FALSE</td>
<td>FALSE</td>
</tr>
</tbody>
</table>

(b)  

\[
\text{IF } \text{InA} = \text{TRUE AND InB} = \text{TRUE} \\
\quad \text{THEN} \\
\quad \quad \text{OutZ} \leftarrow \text{FALSE} \\
\quad \text{ELSE} \\
\quad \quad \text{OutZ} \leftarrow \text{TRUE} \\
\text{ENDIF}
\]

Mark as follows  

Structure: IF - THEN - ELSE - ENDIF  

Condition: InA = TRUE AND InB = TRUE  

Logic:  

\[
\{ \text{OutZ} \leftarrow \text{FALSE (when condition true)} \} \\
\{ \text{OutZ} \leftarrow \text{TRUE (when condition false)} \}
\]

Alternative answer (worth 3 marks):  

\[
\text{OutZ} \leftarrow \text{NOT(InA AND InB)} \\
\text{OutZ} \leftarrow \text{NOT InA OR NOT InB}
\]
Start and Stop/End to Score

Either Yes or No labelled

INPUT

Purchase

Purchase > 1000
Yes

No

Purchase > 500
Yes

No

DiscountRate ← 0

DiscountRate ← 0.05

DiscountRate ← 0.10

Paid ← Purchase * (1 - DiscountRate)

OUTPUT

Paid

Stop

[Total: 6]
5  (a)  

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>YearCount</td>
<td>INTEGER</td>
<td>Loop counter /// Age of the car</td>
</tr>
<tr>
<td>PurchasePrice</td>
<td>INTEGER</td>
<td>Purchase price of the car</td>
</tr>
<tr>
<td>CurrentValue</td>
<td>REAL // CURRENCY</td>
<td>The changing depreciated value</td>
</tr>
<tr>
<td></td>
<td>Allow: SINGLE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Refuse: DOUBLE</td>
<td></td>
</tr>
</tbody>
</table>

Must have correct identifier + Data type + Description to score

(b)  

OUTPUT "Enter Purchase price"
INPUT PurchasePrice
CurrentValue ← PurchasePrice
YearCount ← 1
WHILE YearCount < 9 AND CurrentValue >= 1000
  Note: Penalise: inclusion of $
  IF YearCount = 1
    THEN
      CurrentValue ← CurrentValue * (1 - 40/100)
  ELSE
    CurrentValue ← CurrentValue * 0.8
  ENDIF
OUTPUT YearCount, CurrentValue
YearCount ← YearCount + 1
ENDWHILE
6  (a) Combination of staff and task number // the pair of numbers // the pair of random numbers [1]
    //there will be duplicates / repeats / some staff tasks will not be generated [1]

    (b) (i) 04 // 03 [1]
    (ii) 27 // 28 [1]
    (iii) 20 [1]
    (iv) 11 / 12 [1]

    (c) (i) Zero [1]
    (ii) Completed <> 60 // NewStaffTask = FALSE [1]
         Allow: Inclusion of the WHILE

    (iii) Determines whether this combination of StaffNum and TaskNum has been completed [1]
         Assigns value TRUE if not already generated [1]
         Flags that this is the first time this staff + task has been selected/to exit the loop [1]
         Outputs the new staff + task number [1]

    (iv) TaskGrid : ARRAY[1:5, 1:12] OF BOOLEAN [MAX 3]
         1 mark                      1 mark [2]

    (d) Pseudocode ...

    SELECT CASE (OF) + ENDCASE using StaffNo [4]
    1 mark    1 mark
    (CASE) 1: StaffName ← "Sadiq" [1]
    (CASE) 2: StaffName ← "Smith" [1]
    (CASE) 3: StaffName ← "Ho" [1]
    (CASE) 4: StaffName ← "Azmah" [1]
    (CASE) 5:StaffName ← "Papadopolis" [1]
    (all four cases ...) [1]

    ENDCASE // ENDS SELECT

Visual Basic
Select Case StaffNo
Case 1
    StaffName = "Sadiq"
Case 2
    StaffName = "Smith"
Case 3
    StaffName = "Ho"
Case 4
    StaffName = "Azmah"
Case 5
    StaffName = "Papadopolis"
End Select
7 (a) (i) CAT
Ignore any opening + closing quotes

(ii) 13

(iii) 83

(iv) 15

(b) *Input of string …
Correct syntax (for both prompt and assignment) *

*Uses MyString identifier*

StringTotal set to 0

*FOR loop:*

FOR - NEXT keywords // (Python) correct indentation
Correct start and /end boundaries // alternative Python syntax

Note: the end boundary must use the language length function/method // alternative Python syntax

Isolate single character number

Followed by the use of Asc (VB) // Ord ( Python)

Assigned to NextNum

Added to StringTotal

Correct syntax for the output of the string and number

[Max 6]

Python …

MyString = input('key in string')

StringTotal = 0
for i in range (0, len(MyString)):
    NextNum = ord(MyString[i])
    StringTotal = StringTotal + NextNum

print(MyString, StringTotal)
Visual Basic...

Dim MyString As String
Dim StringTotal As Integer
Dim i As Integer
Dim NextNum As Integer

Console.Write("key in string")
MyString = Console.ReadLine
StringTotal = 0
For i = 1 To Len(MyString) // MyString.Length
    NextNum = Asc(Mid(MyString, i, 1))
    StringTotal = StringTotal + NextNum
Next

Console.WriteLine(MyString & " " & Str(StringTotal))

Pascal ...

VAR MyString : String ;
VAR StringTotal : Integer ;
VAR i : Integer ;
VAR NextNum : Integer ;
VAR SingleChar : Char ;

begin
    Writeln('key in string');
    readln(MyString) ;
    StringTotal := 0 ;

    For i := 1 To Length(MyString) do 
    begin
        SingleChar := MyString[i] ;
        NextNum := Ord(SingleChar) ;
        StringTotal := StringTotal + NextNum ;
    end ;

    WriteLn(MyString, StringTotal) ;

    ReadLn() ;
End.

(c) Used to provide an integrity/verification check [1]
Used as a checksum [1]
The total can be recalculated by the receiving software [1]
If any of the characters have been incorrectly transmitted the recalculated total and transmitted total will not match [1]

[MAX 2]
8  (a)  \( r \) 
Ignore inclusion of any quotes  

(b)  
(i)  2  
Ignore inclusion of any quotes for part (i), (ii) and (iii)  
(ii)  +  
(iii)  7  

(c)  
(i)  

<table>
<thead>
<tr>
<th>N1</th>
<th>N2</th>
<th>N3</th>
<th>N4</th>
<th>BottomAnswer</th>
<th>Op</th>
<th>TopAnswer</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>3</td>
<td>8</td>
<td>40</td>
<td>-</td>
<td>1</td>
<td>1/40</td>
</tr>
</tbody>
</table>

(ii)  

<table>
<thead>
<tr>
<th>N1</th>
<th>N2</th>
<th>N3</th>
<th>N4</th>
<th>BottomAnswer</th>
<th>Op</th>
<th>TopAnswer</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>16</td>
<td>+</td>
<td>16</td>
<td>1</td>
</tr>
</tbody>
</table>

(iii)  

<table>
<thead>
<tr>
<th>N1</th>
<th>N2</th>
<th>N3</th>
<th>N4</th>
<th>BottomAnswer</th>
<th>Op</th>
<th>TopAnswer</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>9</td>
<td>2</td>
<td>3</td>
<td>27</td>
<td>+</td>
<td>39</td>
<td>12/27</td>
</tr>
</tbody>
</table>
(d) (i) Adaptive (maintenance) [1]

(ii) Allow more than two fractions to be added [1]
     Numerator/denominator more than 1 digit [1]
     Multiply and division also possible [1]
     Allow brackets [1]
     Give answer as decimal number [1]
     Lowest possible denominator [1]
     Trap any fraction which has a zero numerator [1]
     Allow the input of vulgar fraction(s) [1]

[MAX 3]