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) 2 [1]

(ii) 7.5
Accept: 7 ½ [1]

(iii) FALSE [1]

(iv) TRUE [1]

(v) ERROR [1]

2 (a)

<table>
<thead>
<tr>
<th>Test Case</th>
<th>P</th>
<th>Q</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(b)

IF P = 1 AND Q = 1
THEN
X ← 1
ELSE
X ← 0
ENDIF
IF P = 0 OR Q = 0
THEN
X ← 0
ELSE
X ← 1
ENDIF

Mark as follows:
Condition: P = 1 AND Q = 1 [1]
Allow &/&& for the operator
Logic: X ← 1 (for TRUE) X ← 0 (for FALSE) [1]

Check carefully for:
- other alternative correct algorithm
- a 'mirror copy' of the question paper algorithm – score 0
START and STOP/END to score

A. CALCULATE points total
In general – accept descriptive assignment statements

Condition + at least one label to score
A. True/Tick

Allow yes/Yes/missing quotes

INPUT Amount(Due)

CALL CalculatePoints(AmountDue)

PointsTotal =
PointsTotal + NewPoints

PointsTotal > 500
YES

OUTPUT “Exchange points?”

INPUT Response

Response = “YES”?
YES

PointsTotal =
PointsTotal - 500

AmountDue = AmountDue - 1

OUTPUT AmountsDue, PointsTotal

STOP

[Max 6]
4 (a) The combination of suit and card number // the ‘pair’ of numbers // the pair of random numbers ....

There will be duplicates/repeats//not all cards will be drawn

(b) (i) 32 // 33

(ii) 27 // 28

(iii) 08

(iv) 12 // 13

(c) 1

(d) DealCount <> 52 // NewCard = FALSE

Allow: Inclusion of the WHILE

(e) Test has the card has already been drawn?

Set value TRUE for this card entry (in the array)/this card

Flags that this is the first time this card has been drawn // decides if another card must be generated

Outputs the new card value

[Max 2]


Allow: parentheses

(g) Pseudocode ...

(SELECT) CASE (OF) CardValue + ENDCASE

(CASE) 1: CardName ← "Ace"

(CASE) 11: CardName ← "Jack"

(CASE) 12: CardName ← "Queen"

(CASE) 13: CardName ← "King"

(OTHERWISE (/ELSE) CardName ← CardValue //

(CASE) 2 TO 10: CardName ← CardValue)

ENDCASE // ENDSWITCH

Note: Must be double quotes present and correct case
Visual Basic
Select Case CardValue
    Case 1
        CardName = "Ace"
    Case 11
        CardName = "Jack"
    Case 12
        CardName = "Queen"
    Case 13
        CardName = "King"
    Case Else // Case 2 to 10
        CardName = Str(CardValue) [4]
End Select
Allow: omission of Str
(a) (i)

- To sort / to order / put in ascending order the items (in the array) [1]
- There were no swaps on the last pass / on pass 4 [1]

<table>
<thead>
<tr>
<th>N</th>
<th>i</th>
<th>j</th>
<th>Temp</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td>16</td>
<td>13</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Terminates at 5 and 4 respectively

Must be:
- Preceded by other entries …
- nothing after the 1, 2 sequence

Note these final values will not be shown on the same row …

The final values are: 2, 7, 8, 11, 13, 16

[8]

(ii) To sort / to order / put in ascending order the items (in the array) [1]

(iii) There were no swaps on the last pass / on pass 4 [1]
Mark as follows:

**INTEGER** × 4  [1]

One mark per description  [4]

6  (a)  (i)  12  [1]

(ii) ‘L’  [1]

Note: quotes are optional – must be upper case L

(b)  (i)  

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InputString</td>
<td>STRING</td>
<td>The string value input by the user</td>
</tr>
<tr>
<td>i</td>
<td>INTEGER</td>
<td>Loop counter // (index) position of an individual character</td>
</tr>
<tr>
<td>j</td>
<td>INTEGER</td>
<td>Number of characters in / length of InputString</td>
</tr>
<tr>
<td>NextChar</td>
<td>CHAR//CHARACTER</td>
<td>(Single) character within InputString / from string input by the user</td>
</tr>
<tr>
<td>NewString</td>
<td>STRING</td>
<td>The string formed/made/created//output Allow: if “by the user” added</td>
</tr>
</tbody>
</table>

Note: Correct (identifier + the data type + description) needed to score
(ii) // main program
INPUT MyString
ChangedString ← RemoveSpaces(MyString)
OUTPUT ChangedString

// function definition
FUNCTION RemoveSpaces(InputString : STRING)
RETURNS STRING
DECLARE i : AS INTEGER
DECLARE j : AS INTEGER
DECLARE NextChar : AS CHAR
DECLARE NewString : AS STRING
NewString = ""
j ← CharacterCount(InputString)
FOR i ← 1 TO j
    NextChar ← OneChar(InputString, i)
    IF NextChar <> " " THEN
        // the & character joins together two strings
        NewString ← NewString & NextChar
    ENDIF
ENDFOR
RETURN NewString // only awarded if follows the previous mark
RemoveSpaces ← NewString
ENDFUNCTION

[Max 7]

7 (a) (i) 165 [1]

(ii) “YES” Quotes optional [1]

(iii) 9 [1]

(iv) 83 [1]
(b) (i) Use of correct identifiers only to score

Declaration/Commenting of variables

MyMessage As String
EncryptString As String
i As Integer
NextNum As Integer
At least two variables correctly documented

Input of string …
Correct syntax (for both prompt and assignment) and …
Uses the MyMessage identifier

EncryptString set to ‘empty string’
Note: Must suggest ‘empty’ string

For loop …
FOR – NEXT keywords // (Python) correct indentation
Correct start/end boundaries
Note: the end boundary must use the language length function/method //alternative Python syntax

Isolate single character
Use of language functions to calculate new number and …. Assigned to NextNum

Conversion of NextNum to a character and concatenated to EncryptString

Correct syntax for output of EncryptString

[MAX 8]

SAMPLE CODE

PYTHON
MyMessage = input("Enter message : ")
EncryptString = ""
for i in range(0, len(MyMessage)) :
    NextNum = ord(MyMessage[i]) + 3
    EncryptString = EncryptString + chr(NextNum)
print(EncryptString)

Alternative solution:
MyMessage = input("Enter message : ")
EncryptString = ""
for NextChar in MyMessage :
    NextNum = ord(NextChar) + 3
    EncryptString = EncryptString + chr(NextNum)
print(EncryptString)
VB
Dim MyMessage, EncryptString As String
Dim NextNum, i As Integer
Console.Write("Enter message : ")
MyMessage = Console.ReadLine()
EncryptString = ""
For i = 1 To Len(MyMessage)
    NextNum = Asc(Mid(MyMessage, i, 1)) + 3
    EncryptString = EncryptString + Chr(NextNum)
Next
Console.WriteLine(EncryptString)

Alternative solution:
Dim MyMessage, EncryptString As String
Dim NextNum, i As Integer
Console.Write("Enter message : ")
MyMessage = Console.ReadLine()
EncryptString = ""
For i = 0 To Len(MyMessage) - 1
    NextNum = Asc(MyMessage.Chars(i)) + 3
    EncryptString = EncryptString + Chr(NextNum)
Next
Console.WriteLine(EncryptString)

PASCAL
var
    MyMessage, EncryptString : string;
    NextNum, i : integer;
begin
    write('Enter message : ');
    readln(MyMessage);
    EncryptString := ';
    for i := 1 to length(MyMessage) do
    begin
        NextNum := ord(MyMessage[i]) + 3;
        EncryptString := EncryptString + chr(NextNum);
    end;
    writeln(EncryptString);
end.

(ii) For each/every character .... [1]
A replacement character is ‘calculated’ from its ASCII value // or by example ... [1]