

QUESTION 7.



- 5 (a) • 1D Array // List [1]
- INTEGER [1]

(b) (i)

x	DayNumber	OUTPUT
0	1	
	2	
1	3	5/6/2015
	4	
2	5	7/6/2015
	6	
3	7	9/6/2015
		3

Note: 'x' and 'output' entries must be on or below the relevant 'DayNumber' entry
Mark as above

[4]



- (ii)
- ... Sales for the first seven days
 - ... the number of days on which the total sales were 10 or over
 - Outputs the corresponding dates
 - Output the final value/total (of x)

(1)
(1)
(1)

(c) (i) 2

[1]

(ii)

Tick Cross	Explanation (if invalid)
X // ✓	2 nd parameter should be CHAR // accept just tick
X	Three parameters/should be 2 parameters
✓	

[3]

(d) OPENFILE "DISCOUNT_DATES" FOR **WRITE** / **WRITING** (1)
 INPUT **NextDate** (1)
 WHILE NextDate <> "XXX"
 INPUT Discount
NextLine = CONCAT(NextDate, " ", Discount) (1)
 WRITEFILE "DISCOUNT_DATES", NextLine
ENDWHILE (1)
 OUTPUT "File now created"
 CLOSEFILE

[4]

(e) (i) Sensible Identifier + Data Type + Description (1 + 1 + 1)

For example:

ThisDate	STRING/DATE	date 'entered by user'
Found	BOOLEAN	flag to indicate ThisDate is 'present in the file'
NextLine	STRING	a single line 'from the text file'
NextDate	STRING/DATE	date 'from next line in the file'
NextDiscount	STRING	the discount value from NextLine
ThisMonth	INTEGER	the month part of the date (input or from file)
MyStreamReader	STREAMREADER	references DISCOUNT_DATES file

Reject 'generic' reserved words

Allow **one** instance variable to store output string(s)

Allow **one** instance of month/day/year number e.g. ThisMonth shown above

[3]



(ii) Mark as follows:

Open file statement	(1)
File read statement for line text – <code>NextLine</code>	(1)
File close statement	(1)

Input of the required date – <code>ThisDate</code>	(1)
--	-----

Isolate <code>NextDate</code> from <code>NextLine</code>	(1)
--	-----

Isolate <code>NextDiscount</code> from <code>NextLine</code>	(1)
--	-----

IF statement comparing the two dates	(1)
--------------------------------------	-----

Uses Boolean variable <code>Found</code> to flag when found	(1)
---	-----

Post/pre condition loop iterate through the file	(1)
--	-----

Test for <u>EOF</u> or 'found'	(1)
--------------------------------	-----

Note: These must follow some correct logic to score ...

Output 'No discount on this date' and Output 'This is a discount date')	(1)
--	-----

Output (when date not found) 'Date not found'	(1)
---	-----

Accept 'any' identifier names	[max 7]
-------------------------------	---------

Page 8	Mark Scheme
	Cambridge International AS/A Level – May/June 2015



APPENDIX Programming Solutions

Question 1 (b) (ii)

Visual Basic ...

```

Dim RaceHours As Integer
Dim RaceMinutes As Integer
Dim RaceSeconds As Integer
Dim RaceTimeInSeconds As Integer
Dim PersonalBest As Integer

Console.Write("Time in hours ... ") : RaceHours = Console.ReadLine
Console.Write("Time in minutes... ") : RaceMinutes = Console.ReadLine
Console.Write("Time in seconds ... ")
RaceSeconds = Console.ReadLine
Console.Write("Personal best in seconds ... ")
PersonalBest = Console.ReadLine
RaceTimeInSeconds = RaceHours*60*60 + RaceMinutes*60 + RaceSeconds
Console.Write(RaceTimeInSeconds)
If RaceTimeInSeconds < PersonalBest Then
    Console.WriteLine("New personal best time")
Else
    If RaceTimeInSeconds = PersonalBest Then
        Console.WriteLine("Equals personal best time")
    Else
        Console.WriteLine("Below personal best")
    End If
End If

```

Python ...

```

# RaceHours          - Integer
# RaceMinutes        - Integer
# RaceSeconds        - Integer
# RaceTimeInSeconds  - Integer
# PersonalBest       - Integer

RaceHours = int(input("Time in hours ... "))
RaceMinutes = int(input("Time in minutes... "))
RaceSeconds = int(input("Time in seconds ... "))
PersonalBest = int(input("Personal best in seconds ... "))

RaceTimeInSeconds = RaceHours*60*60 + RaceMinutes*60 + RaceSeconds

if RaceTimeInSeconds < PersonalBest:
    print("New personal best time")
elif RaceTimeInSeconds == PersonalBest:
    print("Equals personal best time")
else:
    print("Below personal best")

```


Page 9	Mark Scheme
	Cambridge International AS/A Level – May/June 2015



Programming Solutions
Question 1 (b) (ii) – contd.

Pascal ...

```

var RaceHours      : Integer ;
var RaceMinutes    : Integer ;
var RaceSeconds    : Integer ;
var RaceTimeInSeconds : Integer ;
var PersonalBestTime : Integer ;

begin
Writeln('Time in hours ... ') ; readln(RaceHours) ;
Writeln('Time in minutes... ') ; readln(RaceMinutes) ;
Writeln('Time in seconds ... ') ;
readln(RaceSeconds) ;
Writeln('Personal best in seconds ... ') ;

Readln(PersonalBest) ;
RaceTimeInSeconds := RaceHours*60*60 + RaceMinutes*60 + RaceSeconds ;
Writeln(RaceTimeInSeconds) ;

If RaceTimeInSeconds < PersonalBestTime Then
    WriteLn('New personal best time')
Else
    If RaceTimeInSeconds = PersonalBest Then
        WriteLn('Equals personal best time')
    Else
        WriteLn('Personal best time is unchanged) ;

Readln;
End

```

Page 10	Mark Scheme	
	Cambridge International AS/A Level – May/June 2015	



Programming Solutions

Question 2 (d)

Visual Basic ...

```

Dim Choice As Integer
Dim NoOfAttempts As Integer
CONST i = 3
Do
    Call DisplayMenu()
    NoOfAttempts = 0
    Do
        Console.Write("Enter choice (1..4) ")
        Choice = Console.ReadLine
        NoOfAttempts = NoOfAttempts + 1
        Loop Until (Choice >= 1 And Choice <= 4) Or NoOfAttempts = i

        If Choice = 1 Then Call ReadFile()
        If Choice = 2 Then Console.WriteLine("Add customer code")
        If Choice = 3 Then Console.WriteLine("Search customer code")
    Loop Until Choice = 4

Sub DisplayMenu()
    Console.WriteLine()
    Console.WriteLine("1. Read customer file")
    Console.WriteLine("2. Add customer")
    Console.WriteLine("3. Search for a customer")
    Console.WriteLine("4. End")
    Console.WriteLine()
End Sub

Sub ReadFile()
    Console.WriteLine("Read file code")
End Sub

```

Python ...

```

def DisplayMenu():
    print()
    print("1. Read customer file")
    print("2. Add customer")
    print("3. Search for a customer")
    print("4. End")
    print()

def ReadFile():
    print("Read file code")

if __name__ == "__main__" :
    # Choice - Integer
    # NoOfAttempts - Integer

    Choice = 0
    while Choice !=4:
        DisplayMenu()

```



```
Choice = int(input"Enter choice (1..4) :")
NoOfAttempts = 1
while (Choice < 1 or Choice >4) and NoOfAttempts < 3:
    Choice = int(input"Enter choice (1..4) :")
    NoOfAttempts = 1
if Choice == 1:
    ReadFile()
elif Choice == 2:
    print("Add customer code")
elif Choice == 3:
    print("Print customer code")
```

Programming Solutions

Question 2 (d) – contd.

Pascal ...

```
var Choice          : Integer ;
var NoOfAttempts    : Integer ;
const i = 3 ;

procedure DisplayMenu ;
begin
    WriteLn();
    WriteLn('1. Read customer file') ;
    WriteLn('2. Add customer') ;
    WriteLn('3. Search for a customer');
    WriteLn('4. End') ;
    WriteLn() ;
End ;

Procedure ReadFile ;
begin
    WriteLn('Read file code');
End ;

begin

repeat
    DisplayMenu() ;
    NoOfAttempts := 0 ;
    repeat
        Writeln('Enter choice (1..4)') ; ReadLn(Choice) ;
        NoOfAttempts := NoOfAttempts + 1 ;
    Until ((Choice >= 1) And (Choice <= 4)) Or (NoOfAttempts = i);

    If Choice = 1 Then ReadFile() ;
    If Choice = 2 Then writeLn('Add customer code');
    If Choice = 3 Then WriteLn('Search customer code') ;
Until Choice = 4 ;

end.
```

Page 12	Mark Scheme
	Cambridge International AS/A Level – May/June 2015



Programming Solutions

Question 5 (ii)

Visual Basic ...

```

Dim ThisDate As String : Dim NextDate As String
Dim FileString As String
Dim Found As Boolean

FileOpen(1, "D:DISCOUNT_DATES.txt", OpenMode.Input)
    or equivalent for a 'StreamReader' solutions

Console.Write("Date to find (DD/MM/YYYY)..")
ThisDate = Console.ReadLine
Found = False
Do
    FileString = LineInput(1)
    NextDate = Left(FileString, 10)
    If NextDate = ThisDate Then
        Found = True
        ' length is 15 when shows TRUE
    If Len(FileString) = 15 Then
        Console.WriteLine("This is a discount date")
    Else
        Console.WriteLine("No discount on this date")
    End If
    End If
Loop Until Found = True Or EOF(1)

FileClose(1)

If Found = False Then
    Console.WriteLine("Date not found")
End If

```

Python ...

```

MyFile = open("c:\DISCOUNT_DATES.txt", "r")
ThisDate = input("Next date ...(XXX to end)")

Found = 0
while Found == 0:
    NextLine = MyFile.readline()
    if not NextLine:
        break

    FileDate = NextLine[0:10]
    DiscountIndicator = NextLine[11:]

    if FileDate == ThisDate:
        Found = 1
        print (ThisDate, DiscountIndicator)

MyFile.close()
if Found == 0:

```

Page 13	Mark Scheme	
	Cambridge International AS/A Level – May/June 2015	



```
print ("This date was not found")
```

Note: Found could be Boolean to give:

```
Found = False
while not Found:
```

Programming Solutions

Question 5 (ii) – contd.

Pascal ...

```
var ThisDate      : String ;
var NextDate      : String ;
var TheFile       : Text   ;
var FileString    : String ;
var Found         : Boolean ;

begin
assign(TheFile, 'k:\DISCOUNT_DATES.txt') ;
reset(TheFile) ;

writeln('Date to find (DD/MM/YYYY)..') ;
readln(ThisDate) ;
Found := False ;

repeat
  readln(TheFile, FileString) ;
  NextDate := copy(FileString,1, 10) ;

  If NextDate = ThisDate then
    begin
      Found := True ;
      { length is 15 when shows TRUE }
      if length(FileString) = 15 then
        writeln('This is a discount date')
      else
        writeln('No discount on this date')
      end ;
    end

until Found = True or EOF(TheFile) ;

close(TheFile) ;

if Found = False then writeln('Date not found') ;
```

QUESTION 8.



- 5 (a) (i) • Set of data items have a common name (1 mark)
• Items are referenced using a subscript/index (1 mark)
• Accept: all data items are of the same data type (1 mark) [max 2]
- (ii) 24 [1]
- (iii) • The total number of amplifiers 'produced' by workers 1, 2 and 3/three workers (1 mark)
• on day 2_ (1 mark) [2]



(b)

WorkerNum	DayNum	WorkerAverage	OUTPUT
1			
2			
3			
1	1		
	2		
	3		
	4		
2	1		
	2		
	3		
	4		
3	1		
	2		
	3		
	4		
1		2.25	
2		2	
3		1.25	INVESTIGATE 3

Worker

1	2
0	
	0
	0
10	
21	
31	
45	
	20
	36
	60
	80
	9
	20
	33
	50

[8]



- (c) (i) WorkerNum : INTEGER (1 mark)
DayNum : INTEGER (1 mark)
WorkerTotal : ARRAY OF INTEGER
(1 mark) (1 mark)
WorkerAverage : REAL (1 mark) [ma]

(ii) PROCEDURE AnalyseProductionData (NumDays : INTEGER, NumWorkers :
INTEGER)

FOR WorkerNum ← 1 TO 3
WorkerTotal [WorkerNum] ← 0
ENDFOR

FOR WorkerNum ← 1 TO 3
FOR DayNum ← 1 TO 4
WorkerTotal [WorkerNum] ← WorkerTotal [WorkerNum] +
ProductionData [WorkerNum, DayNum]

ENDFOR
ENDFOR

FOR WorkerNum ← 1 TO 3
WorkerAverage = WorkerTotal [WorkerNum] / (4)*
DailyHoursWorked [WorkerNum]
IF WorkerAverage < 2
THEN
OUTPUT "Investigate" WorkerNum
ENDIF
ENDFOR

ENDPROCEDURE

Mark as follows:

All '3's changed to NumWorkers

All '4's changed to NumDays

WorkerAverage '4' changed to NumDays [3]

(iii) (CALL) AnalyseProductionData (7, 13) [1]

QUESTION 9.



8 (a) r
Ignore inclusion of any quotes

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

(ii) + [1]

(iii) 7 [1]

(c) (i)

N1	N2	N3	N4	BottomAnswer	Op	TopAnswer	OUTPUT
2	5	3	8	40	-	1	1/40

[2]

(ii)

N1	N2	N3	N4	BottomAnswer	Op	TopAnswer	OUTPUT
3	4	1	4	16	+	16	1

[2]

(iii)

N1	N2	N3	N4	BottomAnswer	Op	TopAnswer	OUTPUT
7	9	2	3	27	+	39	
						12	
							1 12/27

[3]



(d) (i) Adaptive (maintenance)

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

[1]
 [1]
 [1]
 [1]
 [1]
 [1]
 [1]

[MAX 3]

QUESTION 10.



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

[1]

Input of string ...

Correct syntax (for both prompt and assignment) and ...

Uses the MyMessage identifier

[1]

EncryptString set to 'empty string'

Note: Must suggest 'empty' string

[1]

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

[1]

[1]

Isolate single character

Use of language functions to calculate new number and

Assigned to NextNum

[1]

[1]

Conversion of NextNum to a character and concatenated
to EncryptString

[1]

Correct syntax for output of EncryptString

[1]

[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.WriteLine("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)
```

Alternatives:

```
GetChar(MyMessage, i)
MyMessage.Substring(i, 1)
```

Allow: Use of InputBox and MsgBox

Alternative solution :

```
Dim MyMessage, EncryptString As String
Dim NextNum, i As Integer
Console.WriteLine("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]

QUESTION 11.



3 (b)(ii): Python

```
StartPos = int(input("Enter start position: "))
NumToChange = int(input("Enter number to change: "))
for n in range(NumToChange) :
    NewChar = input("Input new value for position: ")
    LookUp[StartPos + n - 1] = NewChar
print(str(NumToChange) + " entries changed")
```

ALTERNATIVE:

```
StartPos = int(input("Enter start position: "))
NumToChange = int(input("Enter number to change: "))
n = 0
while n < NumToChange :
    NewChar = input("Input new value for position: ")
    LookUp[StartPos + n] = NewChar
    n = n + 1
print(str(NumToChange) + " entries changed")
```

Page 13	Mark Scheme	
	Cambridge International AS/A Level – May/June 2016	



5 (b): VB.NET

A StreamWriter() solution:

```
Sub InputData()
    Dim CDTitle, CDArtist, CDLocation As String
    Dim FileHandle As IO.StreamWriter
    FileHandle = New IO.StreamWriter("MyMusic.txt") ("MyMusic.txt")
    Console.WriteLine("Input CD Title: ")
    CDTitle = Console.ReadLine()
    Do Until CDTitle = "##"
        Console.WriteLine("Input CD Artist: ")
        CDArtist = Console.ReadLine()
        Console.WriteLine("Input CD Location: ")
        CDLocation = Console.ReadLine()
        FileHandle.WriteLine(CDTitle & ":" & CDArtist & ":" & CDLocation)
        Console.WriteLine("Input CD Title: ")
        CDTitle = Console.ReadLine()
    Loop
    FileHandle.Close()
End Sub
```

A legacy FileOpen() solution:

```
Sub InputData()
    Dim CDTitle, CDArtist, CDLocation As String
    FileOpen(1, "MyMusic", OpenMode.Output)
    Console.WriteLine("Input CD Title: ")
    CDTitle = Console.ReadLine()
    Do Until CDTitle = "##"
        Console.WriteLine("Input CD Artist: ")
        CDArtist = Console.ReadLine()
        Console.WriteLine("Input CD Location: ")
        CDLocation = Console.ReadLine()
        Print(1, CDTitle & ":" & CDArtist & ":" & CDLocation)
        Console.WriteLine("Input CD Title: ")
        CDTitle = Console.ReadLine()
    Loop
    FileClose(1)
End Sub
```

Page 14	Mark Scheme	
	Cambridge International AS/A Level – May/June 2016	



5 (b): Pascal

```

procedure InputData;
var
    CDTitle, CDArtist, CDLocation : string;
    CDFile : Textfile;
begin
    assign(CDFile, 'MyMusic');
    rewrite(CDFile);
    writeln('Input CD Title: ');
    readln(CDTitle);
    while (CDTitle <> '##') do
    begin
        writeln('Input CD Artist: ');
        readln(CDArtist);
        writeln('Input CD Location: ');
        readln(CDLocation);
        writeln(CDFile, CDTitle + ':' + CDArtist + ':' + CDLocation);
        writeln('Input CD Title: ');
        readln(CDTitle);
    end;
    close(CDFile);
end;

```

5 (b): Python

```

def InputData() :
    #CDTitle String (or CDTitle = "")
    #CDArtist String (or CDArtist = "")
    #CDLocation String (or CDLocation = "")

    FileHandle = open("MyMusic", "w")
    CDTitle = input("Input CD Title: ")
    while CDTitle != "##" :
        CDArtist = input("Input CD Artist: ")
        CDLocation = input("Input CD location: ")
        FileHandle.write(CDTitle + ":" + CDArtist + ":" + CDLocation)
        CDTitle = input("Input CD Title: ")
    FileHandle.close()

```


QUESTION 2.



3 (a)

[11]

```
FUNCTION Decrypt (Lookup : ARRAY, CipherChar : CHAR) RETURNS CHAR
  DECLARE Found           : BOOLEAN
  DECLARE Index           : INTEGER
  DECLARE OriginalChar    : CHAR

  Index ← 1                //start with first element in the array //
                           assign the start index

  Found ← FALSE
  //now search for CipherChar in Lookup:
  WHILE Found = FALSE // Found <> TRUE // NOT Found
    // compare CipherChar with this array element:
    IF Lookup[Index] = CipherChar
      THEN
        Found ← TRUE           // Set the flag
      ELSE
        Index ← Index + 1      //Move to next array element
      ENDIF
    ENDWHILE
  //dropped out of loop so must have found CipherChar:
  OriginalChar ← CHR(Index)    // convert Index to original character
  RETURN OriginalChar

ENDFUNCTION
```

One mark for each part-statement (shown underlined and bold)

Page 5	Mark Scheme	
	Cambridge International AS/A Level – May/June 2016	



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

```

INPUT StartIndex
INPUT NumberToOutput } ①

FOR Index ← StartIndex to StartIndex + NumberToOutput - 1
    OriginalChar ← CHR(Index) ③
    CipherChar ← Lookup[Index] ④
    OUTPUT ("Index " & Index & ": Character " & OriginalChar & ⑤
           " has substitute character " & CipherChar) ⑥
ENDFOR
  
```

② {

Mark points as circled, descriptions as below:

1. Two INPUT statements
2. Working loop using Index (allow alternative solutions including separate loop counter)
3. Assignment (using correct values of Index or other variable)
4. Assignment (using correct values of Index or other variable)
5. One mark for OUTPUT of a string combining text and variables

QUESTION 3.



end.

4(e): Python

```
def ProductCodeSearch(SearchCode):  
    # list indexes start at zero  
    i = 0  
    Found = "no"  
    while not(i == 1001 or Found == "yes"):  
        if SearchCode == PCode[i]:  
            Found = "yes"
```

QUESTION 4.



- | | | | | |
|----------|------------|--------------|----------------|-----|
| 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)

i	j	NextChar	NextNumberString	Numbers		
				1	2	3
1	1	'2'				
			""			
			"2"			
	2	'3'	"23"			
	3	'*'		23		
2	4	'7'	""			
			"7"			
	5	'3'	"73"			
	6	'1'				
	7	'*'	"731"		731	
3	8	'5'	""			
	9	'*'	"5"			5
4	10	'#'				

One mark for each of columns 1 to 4

One mark for numbers 2 & 3 as shown in box

[5]

(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)

[Max. 2]

QUESTION 5.



Question	Answer	
2(a)	Stepwise refinement	
2(b)	1 mark for first 2 data types – String 1 mark for last 2 data types – Boolean 1 mark for each description: PasswordInput Stores password <u>entered</u> UserIDFound True if user ID found in the <u>file</u> PasswordValid True if password entered matches password from <u>file</u> //Input password matches stored password	
2(c)	1. LOOP through the file until EOF... 2. ...OR UserIdInput is found 3. READ text line from Password.txt file in a loop 4. SPLIT into UserID and password in a loop 5. IF UserIdInput matches UserID from file THEN in a loop 6. SET UserIDFound to TRUE in a loop 7. IF UserIDFound = TRUE AND PasswordInput matches value from file THEN 8. Set PasswordValid to TRUE Mark as follows: 1 mark per functional equivalent of each numbered statement.	8

QUESTION 6.

Cambridge
International
AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level



COMPUTER SCIENCE

9608/22

Paper 2 Written Paper

May/June 2017

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 2017 series for most Cambridge IGCSE[®], Cambridge International A and AS Level and Cambridge Pre-U components, and some Cambridge O Level components.

© IGCSE is a registered trademark.

This document consists of **8** printed pages.





Question	Answer																													
1(a)	<table><tr><th>Item</th><th>Statement</th><th>Input</th><th>Process</th><th>Output</th></tr><tr><td>1</td><td>SomeChars = "Hello World"</td><td></td><td>✓</td><td></td></tr><tr><td>2</td><td>OUTPUT RIGHT(String1,5)</td><td></td><td>✓</td><td>✓</td></tr><tr><td>3</td><td>READFILE (MyFile, String2)</td><td>✓</td><td></td><td></td></tr><tr><td>4</td><td>WRITEFILE (MyFile, "Data is " & String2)</td><td></td><td>✓</td><td>✓</td></tr></table>					Item	Statement	Input	Process	Output	1	SomeChars = "Hello World"		✓		2	OUTPUT RIGHT(String1,5)		✓	✓	3	READFILE (MyFile, String2)	✓			4	WRITEFILE (MyFile, "Data is " & String2)		✓	✓
	Item	Statement	Input	Process	Output																									
	1	SomeChars = "Hello World"		✓																										
	2	OUTPUT RIGHT(String1,5)		✓	✓																									
	3	READFILE (MyFile, String2)	✓																											
4	WRITEFILE (MyFile, "Data is " & String2)		✓	✓																										
Mark as follows:																														
Row 1 as shown																														
Row 2 no marks if tick in Input column, otherwise 1 mark per tick																														
Row 3 as shown																														
Row 4 no marks if tick in Input column, otherwise 1 mark per tick																														
1(b)(i)	- Integer / Real / Single / Double / Floating Point / Float - Boolean				2																									
1(b)(ii)	<table><tr><th>Expression</th><th>Evaluates to</th></tr><tr><td>(FlagA AND FlagB) OR FlagC</td><td>TRUE</td></tr><tr><td>FlagA AND (FlagB OR FlagC)</td><td>TRUE</td></tr><tr><td>(NOT FlagA) OR (NOT FlagC)</td><td>FALSE</td></tr></table>				Expression	Evaluates to	(FlagA AND FlagB) OR FlagC	TRUE	FlagA AND (FlagB OR FlagC)	TRUE	(NOT FlagA) OR (NOT FlagC)	FALSE	3																	
Expression	Evaluates to																													
(FlagA AND FlagB) OR FlagC	TRUE																													
FlagA AND (FlagB OR FlagC)	TRUE																													
(NOT FlagA) OR (NOT FlagC)	FALSE																													
1 mark per answer																														
1(c)	MyCount ← 101 REPEAT OUTPUT MyCount MyCount ← MyCount + 2 UNTIL MyCount > 199 1 mark for each of the following: - Counter initialisation - Repeat ... Until loop - Method for choosing (correct range of) odd numbers - Output all odd numbers in the range Note: Counter variable name must be consistent				4																									



Question	Answer	
2(a)	- to <u>increase the level of detail of an algorithm / design...</u> // breaking down a <u>problem / module / task</u> into smaller parts... ...from which the task may be <u>programmed</u> 1 mark per underlined phrase or equivalent	
2(b)	1 mark for first 3 data types – String 1 mark for last data type – Boolean 1 mark for each description: FileUserID Stores (User) ID from <u>file</u> FilePreferredName Stores (preferred) name from <u>file</u> IDFoundFlag True if (User) ID found in <u>file</u> // False if (User) ID not found in <u>file</u> // If SearchUserID matches FileUserID	5
2(c)	- LOOP through the file until EOF()... - OR SearchUserId is found - READ text line from UserNames.txt file in a loop - EXTRACT FileUserID in a loop - IF SearchUserId matches FileUserID THEN in a loop - SET FilePreferredName to the name from the file - Check if User ID found not in a loop - OUTPUT appropriate message for both conditions 1 mark per functional equivalent of each numbered statement.	Max 8



Question	Answer
3	<pre> FUNCTION ExCamel (<u>InString</u>: <u>STRING</u>) RETURNS <u>STRING</u> DECLARE NextChar : <u>CHAR</u> DECLARE <u>OutString</u> : <u>STRING</u> DECLARE n : <u>INTEGER</u> <u>OutString</u> ← "" // initialise the return string // loop through InString to produce OutString FOR n ← 1 TO <u>LENGTH(InString)</u> // from first to last NextChar ← <u>MID(InString, n, 1)</u> // get next character IF <u>NextChar >= 'A' AND NextChar <= 'Z'</u> // check if upper case // <u>NextChar = UCASE(NextChar)</u> THEN IF n > 1 // if not first character THEN <u>OutString</u> ← <u>OutString & " "</u> // add space to OutString ENDIF <u>NextChar</u> ← <u>LCASE(NextChar)</u> // make NextChar lower case ENDIF <u>OutString</u> ← <u>OutString & NextChar</u> // add Nextchar to OutString ENDFOR <u>RETURN OutString</u> // return value ENDFUNCTION </pre> 1 mark per underlined word / expression



Question	Answer											
4(a)	• Functions • Procedures • Global / Local variables 1 mark per item											
4(b)	<table><tr><th>Name of parameter passing method</th><th>Value output</th><th>Explanation</th></tr><tr><td>(Call) by reference</td><td>5</td><td> • The <u>address of</u> the variable is passed. • <u>Original value is changed</u> when parameter changed in called module. </td></tr><tr><td>(Call) by value</td><td>4</td><td> • A <u>copy of</u> the variable itself is passed. • <u>Original value not changed</u> when parameter changed in called module. </td></tr></table> Mark as follows: • 1 mark for each name and value • 1 mark per bullet in explanation			Name of parameter passing method	Value output	Explanation	(Call) by reference	5	• The <u>address of</u> the variable is passed. • <u>Original value is changed</u> when parameter changed in called module.	(Call) by value	4	• A <u>copy of</u> the variable itself is passed. • <u>Original value not changed</u> when parameter changed in called module.
Name of parameter passing method	Value output	Explanation										
(Call) by reference	5	• The <u>address of</u> the variable is passed. • <u>Original value is changed</u> when parameter changed in called module.										
(Call) by value	4	• A <u>copy of</u> the variable itself is passed. • <u>Original value not changed</u> when parameter changed in called module.										



6

Question	Answer	Marks
5(a)(i)	• Any character <u>except</u> colon, space or any alpha-numeric • Reason: character is not in the login information strings	2
5(a)(ii)	DECLARE <u>LogArray</u> : ARRAY[1 : 20] OF <u>STRING</u> 1 mark per underline	2



Question	Answer
5(b)	Pseudocode solution included here for development and clarification of mark scheme. Programming language example solutions appear in the Appendix. <pre> PROCEDURE LogEvents() DECLARE FileData : STRING DECLARE ArrayIndex : INTEGER OPENFILE "LoginFile.txt" FOR APPEND FOR ArrayIndex ← 1 TO 20 // IF LogArray[ArrayIndex] <> "*****" THEN FileData ← LogArray[ArrayIndex] WRITEFILE ("LoginFile.txt", FileData) ENDIF ENDFOR CLOSEFILE("LoginFile.txt") ENDPROCEDURE </pre> 1 mark for each of the following: 1. Procedure heading and ending 2. Declare <code>ArrayIndex</code> as integer // commented in python 3. Open file 'LoginFile' for append 4. Correct loop 5. extract data from array in a loop 6. check for unused element in a loop 7. write data to file in a loop 8. Close the file outside the loop



Question	Answer
6(a)	Pseudocode solution included here for development and clarification of mark scheme. Programming language example solutions appear in the Appendix. <pre> FUNCTION ValidateRegistration(Registration : STRING) RETURNS BOOLEAN DECLARE UCaseChar, NumChar : INTEGER DECLARE NextChar : CHAR DECLARE ReturnFlag : BOOLEAN DECLARE n : INTEGER ReturnFlag ← TRUE ValidateRegistration ← True IF LEN(Registration) < 6 OR LEN(Registration) > 9 //check length THEN ReturnFlag ← False ELSE FOR n ← 1 TO 3 //check for 3 upper case alpha NextChar ← MID(Registration, n, 1) IF NextChar < 'A' AND NextChar > 'Z' THEN ReturnFlag ← False ENDIF ENDFOR FOR n ← 4 TO 5 //check for 2 numeric NextChar ← MID(Registration, n, 1) IF NextChar < '0' AND NextChar > '9' THEN ReturnFlag ← False ENDIF ENDFOR FOR n ← 6 TO LEN(Registration) //check remaining characters NextChar ← MID(Registration, n, 1) IF NextChar < 'A' AND NextChar > 'Z' THEN ReturnFlag ← False ENDIF ENDFOR ENDIF RETURN (ReturnFlag) ENDFUNCTION </pre>



Question	Answer	
6(a)	1 mark for each of the following: 1. Correct Function heading and ending 2. Check for correct length 3. Extract first three characters 4. Check first three characters are capitals 5. Extract characters four and five 6. Check characters four and five are numeric 7. Extract remaining characters 8. Check remaining characters are capitals 9. Combine all four tests results into a single Boolean value 10. Return a Boolean value	
6(b)	String1: (for example, "ABC12XYZ") One mark for a valid string having: • Correct length (between 6 and 9 characters) • 3 capital letters followed by... • 2 numeric characters followed by... • between 1 and 4 capital letters String2 to String5: 1 mark for each string and explanation (testing different rules of the function) Test strings breaking one different rules: • Incorrect length • With incorrect number of capital letters at the start • With non-numeric characters in positions 4 and 5 • With incorrect number of capital letters at the end • Containing an invalid character (not alpha-numeric)	5

QUESTION 7.



Cambridge Assessment International Education
Cambridge International Advanced Subsidiary and Advanced Level



COMPUTER SCIENCE

9608/21

Paper 2 Written Paper

October/November 2017

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 International will not enter into discussions about these mark schemes.

Cambridge International is publishing the mark schemes for the October/November 2017 series for most Cambridge IGCSE[®], Cambridge International A and AS Level components and some Cambridge O Level components.

© IGCSE is a registered trademark.

This document consists of **12** printed pages.





Question	Answer															
1(a)(i)	<table><tr><th>Data value</th><th>Data type</th></tr><tr><td>27</td><td>INTEGER</td></tr><tr><td>"27"</td><td>STRING</td></tr><tr><td>"27.3"</td><td>STRING</td></tr><tr><td>TRUE</td><td>BOOLEAN</td></tr><tr><td>27/3/2015</td><td>DATE // DATETIME</td></tr><tr><td>27.3</td><td>REAL</td></tr></table> One mark for each data type Mark first data type given in each case	Data value	Data type	27	INTEGER	"27"	STRING	"27.3"	STRING	TRUE	BOOLEAN	27/3/2015	DATE // DATETIME	27.3	REAL	
Data value	Data type															
27	INTEGER															
"27"	STRING															
"27.3"	STRING															
TRUE	BOOLEAN															
27/3/2015	DATE // DATETIME															
27.3	REAL															
1(a)(ii)	1D Array // 1DList	2														
1(a)(iii)	- Each character is represented by an <u>unique</u> / <u>corresponding</u> - binary code / integer / value	2														
1(b)	- When a section of code would be repeated - When a piece of code is needed to perform a specific task - To support modular programming / step wise refinement - Easier to debug / maintain - Built-in / library routines are tried and tested One mark per answer	Max 2														
1(c)	<pre>CASE OF MyVar 1: CALL Proc1() 2: CALL Proc2() 3: CALL Proc3() OTHERWISE OUTPUT "Error" ENDCASE</pre> One mark for: - First line and ENDCASE - All clauses for 1, 2 and 3 'OTHERWISE' clause - OUTPUT statement	4														



Question	Answer
1(d)	Ability to recognise: • selection statement • iteration statement • assignment statements • data declarations / structures / data types / use of variables or objects • modular structure / functions / procedures / subroutines • subroutine parameters • Specific types of statement, e.g. Input, Output, File operations • Code format • Operators Mark as follows: Any two from above, or valid alternative Accept by example

Question	Answer	Marks																																										
2(a)	<table><tr><th>StartNumber</th><th>EndNumber</th><th>Divisor</th><th>NumberFound</th><th>Number</th><th>Remainder</th><th>Output</th></tr><tr><td>11</td><td>13</td><td>2</td><td>0</td><td>11</td><td>1</td><td></td></tr><tr><td></td><td></td><td></td><td></td><td>12</td><td>0</td><td>12</td></tr><tr><td></td><td></td><td></td><td>1</td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td>13</td><td>1</td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td>Count: 1</td></tr></table> One mark for correct Number column One mark for correct Remainder column One mark for correct Output	StartNumber	EndNumber	Divisor	NumberFound	Number	Remainder	Output	11	13	2	0	11	1						12	0	12				1								13	1								Count: 1	3
StartNumber	EndNumber	Divisor	NumberFound	Number	Remainder	Output																																						
11	13	2	0	11	1																																							
				12	0	12																																						
			1																																									
				13	1																																							
						Count: 1																																						
2(b)	Mark as follows: - For a (given) range of values - Counts the number of times one number (numerator) is an exact divisor of the other - Outputs each numerator (only) - Outputs the count Accept by example No mark for: ...calculate the remainder ...add one to NumberFound	3																																										



Question	Answer
2(c)	<pre> graph TD Start([START]) --> Input[/INPUT StartNumber, EndNumber, Divisor/] Input --> Init1[NumberFound ← 0] Init1 --> Init2[Number ← StartNumber] Init2 --> LoopStart(()) LoopStart --> Mod[Remainder ← MODULUS(Number, Divisor)] Mod --> Div{Remainder = 0?} Div -- YES --> Out1[/OUTPUT Number/] Out1 --> IncFound[Increment NumberFound] IncFound --> LoopStart Div -- NO --> LoopStart LoopStart --> EndCheck{Number = EndNumber?} EndCheck -- YES --> Out2[/OUTPUT "Count: " & NumberFound/] Out2 --> Stop([STOP]) EndCheck -- NO --> IncNum[Increment Number] IncNum --> LoopStart </pre> The flowchart describes the process of counting numbers within a specific range that are divisible by a given divisor. It begins with a 'START' terminal, followed by an input block for 'StartNumber', 'EndNumber', and 'Divisor'. The process then initializes 'NumberFound' to 0 and sets 'Number' to 'StartNumber'. A loop begins where the remainder of 'Number' divided by 'Divisor' is calculated. If the remainder is 0, the current 'Number' is output, and 'NumberFound' is incremented. In either case (remainder 0 or not), the loop proceeds to check if 'Number' has reached 'EndNumber'. If it has, the final count is output and the process stops. If not, 'Number' is incremented, and the loop repeats from the modulus calculation step.



Question	Answer
2(c)	Mark as follows: • One mark for <code>START</code> and <code>STOP / END</code> • One mark for bracketed pair • One mark for each of other labelled boxes (shape must be correct for decision box) Decision box outputs must have two outputs and at least one label (<code>Yes / No</code>) Different statement categories should not appear in the same symbol (e.g. assignment and I/O) No mark for symbol (or pair) if parent missing or logically incorrect (except for <code>START/END</code>) Full marks should be awarded for functionally equivalent solutions.



Question	Answer
3(a)	<pre> PROCEDURE BubbleSort DECLARE Temp : STRING DECLARE FirstID, SecondID : INTEGER DECLARE NoSwaps : BOOLEAN DECLARE Boundary : INTEGER Declare J : INTEGER Boundary ← 99 REPEAT NoSwaps ← TRUE FOR J ← 1 TO Boundary FirstID ← UserNameArray[J] SecondID ← UserNameArray[J + 1] IF FirstID > SecondID THEN Temp ← UserNameArray[J] UserNameArray[J] ← UserNameArray[J + 1] UserNameArray[J + 1] ← Temp NoSwaps ← FALSE ENDIF ENDFOR Boundary ← Boundary - 1 UNTIL NoSwaps = TRUE ENDPROCEDURE </pre> Mark as follows: 1. Procedure heading and ending (allow array as input parameter) 2. Variable declaration for counter / index (integer) or temp (string) 3. Outer working loop 4. Inner loop with suitable range 5. Correct comparison in a loop 6. Correct swap of complete array element in a loop 7. Set flag to indicate swap in inner loop and resetting in outer loop 8. Reducing Boundary in a loop



Question	Answer
3(b)	Pseudocode solution included here for development and clarification of mark scheme. Programming language example solutions appear in the Appendix. <pre> PROCEDURE FindRepeats DECLARE i, RepeatCount: INTEGER DECLARE FirstID, SecondID: STRING RepeatCount ← 0 FOR i ← 2 TO 100 FirstID ← LEFT(UsernameArray[i - 1], 6) SecondID ← LEFT(UsernameArray[i], 6) IF FirstID = SecondID THEN RepeatCount ← RepeatCount + 1 OUTPUT(UsernameArray[i]) ENDIF ENDFOR IF RepeatCount = 0 THEN OUTPUT "The array contains no repeated UserIDs" ELSE OUTPUT "There are " & RepeatCount & " repeated userIDs" ENDIF ENDPROCEDURE </pre> Mark as follows (all must be correct syntax for chosen language): 1. Procedure heading and ending 2. Variable declaration for INTEGER (comment in Python) and initialisation for RepeatCount (or equivalent name) 3. Loop 4. Extraction of UserID in a loop 5. Correct comparison of consecutive elements... in a loop 6. ...output correct array element (NOT original, only duplicates) in a loop 7. increment RepeatCount following a comparison in a loop 8. Correct conditional statement checking RepeatCount (or equivalent) and then two correct final OUTPUT statements



Question	Answer	
3(c)(i)	• Problem definition • Design • Coding / programming • Testing • Documentation • Implementation • Maintenance	
3(c)(ii)	<u>Integrated Development Environment</u> or a suitable description	1
3(c)(iii)	Examples include: • context sensitive prompts • (dynamic) syntax checking • use of colours to highlight key words / pretty printing • Formatting • Single-stepping • Breakpoints • Report / watch window • (UML) modelling • Compiler/interpreter • Text editor	Max 2
3(c)(iv)	Run-time	1

Question	Answer	Marks												
4(a)	<table><thead><tr><th>Value</th><th>Formatted String</th></tr></thead><tbody><tr><td>1327.5</td><td>"□ 1327.50"</td></tr><tr><td>1234</td><td>"□ 1234.00"</td></tr><tr><td>7.456</td><td>"□□□ 07.45"</td></tr></tbody></table> Leading spaces must be present	Value	Formatted String	1327.5	"□ 1327.50"	1234	"□ 1234.00"	7.456	"□□□ 07.45"	2				
Value	Formatted String													
1327.5	"□ 1327.50"													
1234	"□ 1234.00"													
7.456	"□□□ 07.45"													
4(b)	<table><thead><tr><th>Value</th><th>Required output</th><th>Mask</th></tr></thead><tbody><tr><td>1234.00</td><td>"1,234.00"</td><td>"0,000.00"</td></tr><tr><td>3445.66</td><td>"£3,445.66"</td><td>"£0,000.00"</td></tr><tr><td>10345.56</td><td>"\$□□10,345"</td><td>"\$##00,000"</td></tr></tbody></table> Currency and 'punctuation' symbols must be as shown	Value	Required output	Mask	1234.00	"1,234.00"	"0,000.00"	3445.66	"£3,445.66"	"£0,000.00"	10345.56	"\$□□10,345"	"\$##00,000"	3
Value	Required output	Mask												
1234.00	"1,234.00"	"0,000.00"												
3445.66	"£3,445.66"	"£0,000.00"												
10345.56	"\$□□10,345"	"\$##00,000"												



Question	Answer	
5(a)	<pre> PROCEDURE MakeNewfile DECLARE OldFileLine : STRING DECLARE NewFileLine : STRING OPENFILE "EmailDetails" FOR READ OPENFILE "NewEmailDetails" FOR WRITE WHILE NOT EOF("EmailDetails") READFILE "EmailDetails", OldFileLine NewFileLine ← "00" & OldFileLine WRITEFILE "NewEmailDetails", NewFileLine ENDWHILE CLOSEFILE "EmailDetails" CLOSEFILE "NewEmailDetails" ENDPROCEDURE </pre> Mark as follows: 1. Variable declaration of <code>STRING</code> for <code>OldFileLine</code> (or equivalent) 2. Open <code>EmailDetails</code> for <code>READ</code> 3. Open <code>NewEmailDetails</code> for <code>WRITE</code> 4. Correct loop checking for <code>EOF (EmailDetails)</code> 5. Reading a line from <code>EmailDetails</code> in a loop 6. Correct concatenation in a loop 7. Writing a line to <code>NewEmailDetails</code> in a loop Closing both files	
5(b)	Invalid string examples: A string with nothing before '@' A string with nothing after '@' A string with 1 or 2 characters after '@' A string with no '@' symbol A string with more than one '@' symbol Explanation Sensible explanation mapping each given string to an individual rule One mark for string One mark for explanation Each rule should be tested once only	6

**Programming Example Solutions****Q3(b): Visual Basic**

```
Sub FindRepeats()  
    Dim Repeats As Integer  
    Dim i As Integer  
    Dim FirstID As String  
    Dim SecondID As String  
  
    Repeats = 0  
    For i = 1 To 99  
        FirstID = Left(UsernameArray(i), 6)  
        SecondID = Left(UsernameArray(i + 1), 6)  
        If FirstID = SecondID Then  
            Console.WriteLine(UsernameArray(i + 1))  
            Repeats = Repeats + 1  
        End If  
    Next i  
  
    If Repeats = 0 Then  
        Console.WriteLine("The array contains no repeated UserIDs")  
    Else  
        Console.WriteLine("There are " & Repeats & " repeated UserIDs")  
    End If  
  
End Sub
```

Alternative:

```
Sub FindRepeats ()  
  
    Dim RepeatCount, i As Integer  
    Dim FirstID, SecondID As String  
  
    RepeatCount = 0  
    For i = 1 to 99  
        FirstID = Left(UsernameArray(i-1),6)  
        SecondID = Left(UsernameArray(i),6)  
        If FirstID = SecondID then  
            Console.WriteLine (UsernameArray(i))  
            RepeatCount = RepeatCount + 1  
        End If  
    Next i  
  
    If RepeatCount = 0 then  
        Console.WriteLine ("The array contains no repeated UserIDs")  
    Else  
        Console.WriteLine ("There are "& RepeatCount & " repeated UserIDs")  
    End If  
  
End Sub
```


**Q3(b): Pascal**

```
procedure FindRepeats ();  
  
var  
    RepeatCount, i : integer;  
    FirstID, SecondID : string;  
  
begin  
    RepeatCount := 0;  
    for i := 1 to 99 do  
        begin  
            FirstID := Copy(UsernameArray[i-1],1,6);  
            SecondID := Copy(UsernameArray[i],1,6);  
            if FirstID = SecondID then  
                begin  
                    writeln (UsernameArray[i]);  
                    RepeatCount := RepeatCount + 1;  
                end;  
            end;  
  
            if RepeatCount = 0 then  
                writeln ('The array contains no repeated UserIDs')  
            else  
                writeln ('There are ', RepeatCount, ' repeated UserIDs')  
            end;  
        end;  
    end;
```

**Q3(b): Python**

```
def FindRepeats():
    #Repeats, i Integer
    #FirstID, SecondID string
    Repeats = 0
    for i in range(0, len(UsernameArray)-1):
        FirstID = (UsernameArray[i])[:6]
        SecondID = (UsernameArray[i+1])[:6]
        if FirstID == SecondID:
            print(UsernameArray[i+1])
            Repeats = Repeats + 1
    if Repeats == 0:
        print("The array contains no repeated UserIDs")
    else:
        print("There are ", Repeats, " repeated UserIDs")
```

Alternative:

```
def FindRepeats ():

    RepeatCount = 0                                ## Defined as an integer

    for i in range (1,100):                        ## depending on next two
lines(0,99) (2,101)
        FirstID = UsernameArray[i-1]              ## Defined as string
        SecondID = UsernameArray[i]               ## Defined as string
        if FirstID[0:6] == SecondID[0:6]:         ## Using split
            print (UsernameArray[i])
            RepeatCount += 1

    if repeatCount == 0:
        print ('The array contains no repeated UserIDs')
    else:
        print ('There are ', RepeatCount, ' repeated UserIDs')
```

QUESTION 8.



Cambridge Assessment International Education
Cambridge International Advanced Subsidiary and Advanced Level



COMPUTER SCIENCE

9608/22

Paper 2 Written Paper

October/November 2017

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 International will not enter into discussions about these mark schemes.

Cambridge International is publishing the mark schemes for the October/November 2017 series for most Cambridge IGCSE[®], Cambridge International A and AS Level components and some Cambridge O Level components.

© IGCSE is a registered trademark.

This document consists of **12** printed pages.



Cambridge Assessment
International Education



Question	Answer															
1(a)(i)	<table><tr><th>Data value</th><th>Data type</th></tr><tr><td>FALSE</td><td>BOOLEAN</td></tr><tr><td>03/03/2013</td><td>DATE // DATETIME</td></tr><tr><td>35</td><td>INTEGER</td></tr><tr><td>"INTEGER"</td><td>STRING</td></tr><tr><td>3.5</td><td>REAL</td></tr><tr><td>"35"</td><td>STRING</td></tr></table> One mark for each data type Mark first data type given in each case	Data value	Data type	FALSE	BOOLEAN	03/03/2013	DATE // DATETIME	35	INTEGER	"INTEGER"	STRING	3.5	REAL	"35"	STRING	
Data value	Data type															
FALSE	BOOLEAN															
03/03/2013	DATE // DATETIME															
35	INTEGER															
"INTEGER"	STRING															
3.5	REAL															
"35"	STRING															
1(a)(ii)	1D Array // 1D List	1														
1(a)(iii)	Ability to recognise: • selection statement • iteration statement • assignment statements • data declarations / structures / data types / use of variables or objects • modular structure / functions / procedures / subroutines • subroutine parameters • Specific types of statement e.g. Input, Output, File operations • Code format • Operators Mark as follows: Any two from above, or valid alternative Accept by example	2														
1(b)(i)	<table><tr><th>Data</th></tr><tr><td></td></tr><tr><td>67 // 0100 0011 // 043h</td></tr><tr><td>65 // 0100 0001 // 041h</td></tr><tr><td>71 // 0100 0111 // 047h</td></tr><tr><td>69 // 0100 0101 // 045h</td></tr><tr><td></td></tr></table> One mark for 67 and 65 One mark for 71 and 69 Accept binary, denary or hex values (hex must be clearly indicated) Max one mark if blank cell anywhere in sequence Ignore any data values before or after the four characters	Data		67 // 0100 0011 // 043h	65 // 0100 0001 // 041h	71 // 0100 0111 // 047h	69 // 0100 0101 // 045h		2							
Data																
67 // 0100 0011 // 043h																
65 // 0100 0001 // 041h																
71 // 0100 0111 // 047h																
69 // 0100 0101 // 045h																



Question	Answer	
1(b)(ii)	• A value representing the number of characters stored at beginning of string - OR • Terminator / special character stored to indicate the end of string One mark for each phrase or equivalent.	
1(c)	Explanation includes: • to pass values to/from the subroutine • to produce re-useable code • to avoid global variables • to allow recursion One mark per answer	Max 3
1(d)(i)	27: MyGrade assigned the value "Fail" 101: Output the text "Invalid Value Entered" Ignore minor spelling mistakes	2
1(d)(ii)	<pre> IF MyMark >= 75 AND MyMark <=100 THEN MyGrade ← "Distinction" ELSE IF MyMark >= 35 AND MyMark <=74 THEN MyGrade ← "Pass" ELSE IF MyMark >= 0 AND MyMark <=34 THEN MyGrade ← "Fail" ELSE OUTPUT "Invalid value entered" ENDIF ENDIF ENDIF ENDIF </pre> One mark for each of: • One correct range test • 'IF' equivalent (nested or not) to three CASE range tests... • ... with three corresponding assignments • Equivalent of CASE OTHERWISE with corresponding OUTPUT statement • Matching (three) ENDIFS (Or one if ELSIFS used) Max 4 if solution doesn't work under all circumstances // is not functionally equivalent to CASE	5



Question	Answer	
2(a)	Mark as follows: - To search for a given value in the array - and output the position in the array if the search value is found - if search value not found then output "Not found"	
2(b)	<pre> graph TD Start([START]) --> Input[/Input SearchValue/] Input --> FoundFlag[FoundFlag ← FALSE] FoundFlag --> Index[Index ← 1] Index --> LoopCond{Index < 101 AND FoundFlag = FALSE?} LoopCond -- YES --> ArrayCheck{ClassName[Index] = SearchVal?} ArrayCheck -- YES --> OutputIndex[/OUTPUT Index/] OutputIndex --> FoundFlagTrue[FoundFlag ← TRUE] FoundFlagTrue --> IncrementIndex[Increment Index] ArrayCheck -- NO --> IncrementIndex LoopCond -- NO --> FoundFlagFalse{FoundFlag = FALSE?} FoundFlagFalse -- YES --> OutputNotFound[/OUTPUT "Not found"/] FoundFlagFalse -- NO --> Stop([STOP]) IncrementIndex --> LoopCond </pre>	9



Question	Answer
2(b)	Mark as follows: - One mark for START and STOP / END - One mark for each bracketed pair - One mark for each of other labelled symbol (decision box shape must be correct) - Allow F/T from incorrect decision symbol Full marks should be awarded for functionally equivalent solutions.

Question	Answer	Marks																																	
3	<table> <tr> <th>Line number</th><th>Error</th><th>Correction</th></tr> <tr> <td>01</td><td>Wrong procedure name – "SortArray"</td><td>PROCEDURE ArraySort</td></tr> <tr> <td>02</td><td>Wrong data type - CHAR</td><td>DECLARE Temp: STRING</td></tr> <tr> <td>03</td><td>Variables undefined</td><td>DECLARE FirstID, SecondID, I, J : INTEGER</td></tr> <tr> <td>04</td><td>Wrong 'Value2' of 100</td><td>FOR I ← 1 TO 99</td></tr> <tr> <td>05</td><td>Wrong range</td><td>FOR J ← 1 TO (100 - I)</td></tr> <tr> <td>06/07</td><td>Wrong function - MODULUS</td><td>Replace MODULUS with TONUM: FirstID ← TONUM(LEFT(Product[J],</td></tr> <tr> <td>06/07</td><td>Wrong value of 6</td><td>Should be 4: FirstID ← TONUM(LEFT(Product[J],</td></tr> <tr> <td>10</td><td>Assigning wrong value to Temp</td><td>Temp ← Product[J]</td></tr> <tr> <td>11</td><td>Assigning wrong value to Product[I]</td><td>Product[J] ← Product[J + 1]</td></tr> <tr> <td>13/14</td><td>Lines reversed</td><td>13 ENDIF 14 ENDFOR</td></tr> </table> One mark for each correct row	Line number	Error	Correction	01	Wrong procedure name – "SortArray"	PROCEDURE ArraySort	02	Wrong data type - CHAR	DECLARE Temp: STRING	03	Variables undefined	DECLARE FirstID, SecondID, I, J : INTEGER	04	Wrong 'Value2' of 100	FOR I ← 1 TO 99	05	Wrong range	FOR J ← 1 TO (100 - I)	06/07	Wrong function - MODULUS	Replace MODULUS with TONUM: FirstID ← TONUM (LEFT(Product[J],	06/07	Wrong value of 6	Should be 4: FirstID ← TONUM(LEFT(Product[J],	10	Assigning wrong value to Temp	Temp ← Product[J]	11	Assigning wrong value to Product[I]	Product[J] ← Product[J + 1]	13/14	Lines reversed	13 ENDIF 14 ENDFOR	Max 8
Line number	Error	Correction																																	
01	Wrong procedure name – "SortArray"	PROCEDURE ArraySort																																	
02	Wrong data type - CHAR	DECLARE Temp: STRING																																	
03	Variables undefined	DECLARE FirstID, SecondID, I, J : INTEGER																																	
04	Wrong 'Value2' of 100	FOR I ← 1 TO 99																																	
05	Wrong range	FOR J ← 1 TO (100 - I)																																	
06/07	Wrong function - MODULUS	Replace MODULUS with TONUM: FirstID ← TONUM (LEFT(Product[J],																																	
06/07	Wrong value of 6	Should be 4: FirstID ← TONUM(LEFT(Product[J],																																	
10	Assigning wrong value to Temp	Temp ← Product[J]																																	
11	Assigning wrong value to Product[I]	Product[J] ← Product[J + 1]																																	
13/14	Lines reversed	13 ENDIF 14 ENDFOR																																	



Question	Answer
4(a)	Pseudocode solution included here for development and clarification of mark scheme. Programming language solutions appear in the Appendix. <pre> PROCEDURE TestRandom (Repetitions : INTEGER) DECLARE Frequency : ARRAY [1 : 10] OF INTEGER DECLARE Expected : REAL / INTEGER //allow either DECLARE NextRandom : INTEGER DECLARE N : INTEGER FOR N ← 1 TO 10 Frequency[N] ← 0 ENDFOR Expected ← INT(Repetitions / 10) CALL RANDOMIZE() //Set random seed FOR N ← 1 TO Repetitions NextRandom ← INT(RND() * 10) + 1 Frequency[NextRandom] ← Frequency[NextRandom] + 1 ENDFOR OUTPUT "The expected frequency is " & Expected OUTPUT "Number Frequency Difference" FOR N ← 1 TO 10 OUTPUT N & " " & Frequency[N] & " " & Frequency[N] - Expected ENDFOR ENDPROCEDURE </pre> Mark as follows: 1. Procedure heading (including parameter) 2. Array declaration – 10 or 11 elements 3. Array declaration – data type 4. Variable declaration for a loop counter (integer) or expected frequency (integer or real) 5. Variable declaration for next random value (For Python solutions, mark points 1 to 4 may be gained by suitable comments) 6. Initialise all elements of array 7. To set all elements to zero 8. Calculate expected frequency



Question	Answer	
4(a)	9. Loop to generate required number of random values 10. Use of relevant <code>RANDOM()</code> function in a loop 11. Generate random integer value in the range 1 to 10 in a loop 12. Increment (array) element in a loop 13. Output expected frequency message not in any loop 14. Output column header text 15. (Loop to) output each row 16. ... including three correct values (spaces optional)	
4(b)	• Single-stepping – to allow program statements to be executed one at a time • Breakpoints – to pause / stop the program at a specific line / statement • Variable / expression watch window – to monitor the value of variables / expressions as the program is run One mark for each Feature (text as above or equivalent) + 1 for meaningful explanation of use in context.	6
4(c)	Program is probably working correctly if: • Header is present giving frequency as 20 • Column headers are present • All rows are present (1 to 10) • The difference is calculated correctly • Output is formatted correctly • Total differences should be zero • Sum of Frequencies should be 200	Max 2



Question	Answer
5	<pre> PROCEDURE RemoveDetails DECLARE FileLine: STRING DECLARE MemberToDelete: STRING OPENFILE "EmailDetails.txt" FOR READ OPENFILE "NewEmailDetails.txt" FOR WRITE INPUT MembershipNumber WHILE NOT EOF("EmailDetails.txt") READFILE "EmailDetails.txt", FileLine IF LEFT(FileLine, 4) <> MembershipNumber THEN WRITEFILE "NewEmailDetails.txt", FileLine ENDIF ENDWHILE CLOSEFILE "EmailDetails.txt" CLOSEFILE "NewEmailDetails.txt" ENDPROCEDURE </pre> Mark as follows: 1. Procedure declaration and end. No parameters. 2. Variable declaration of <code>STRING</code> for variable <code>FileLine</code> (or similar) 3. Input the <code>MembershipNumber</code> of the person who has left 4. Open <code>EmailDetails</code> for <code>READ</code> 5. Open <code>NewEmailDetails</code> for <code>WRITE</code> 6. Correct loop checking for <code>EOF (EmailDetails)</code> 7. Reading a line from <code>EmailDetails.txt</code> in a loop 8. Correct check for <code>MemberToDelete</code> in a loop 9. Writing a line to <code>NewEmailDetails.txt</code> in a loop 10. Closing both files (not in a loop)



Appendix - Program Code Example Solutions

Q4 (a): Visual Basic

```

Dim random As New Random()

Sub TestRandom(ByVal repetitions As Integer)
    Dim randinrange As Integer
    Dim i As Integer
    Dim num(1 To 10) As Integer
    Dim freq As Integer
    Dim difference As Integer

    freq = repetitions / 10    'calculate expected frequency

    For i = 1 To 10            'initialise array to store total frequencies
        num(i) = 0
    Next i

    For i = 1 To repetitions 'generate random numbers & increment
        appropriate freq
        randinrange = random.Next(1, 11)
        num(randinrange) = num(randinrange) + 1
    Next i

    Console.WriteLine("The expected frequency is " & freq)    'report header
    Console.WriteLine("Number    Frequency    Difference")    'column headers

    For i = 1 To 10    'calc & display difference between expected and actual
        freq
        difference = num(i) - freq
        Console.WriteLine(i & "    " & num(i) & "    " & difference)
    Next i

End Sub

```

Other possible ways of calculating a random number in VB include:

```

    randinrange = CInt(Math.Floor((upperbound - lowerbound + 1) * Rnd())) +
lowerbound

    randinrange = math.round((Rnd()*9)+1)

    randinrange = CInt(Math.Ceiling(Rnd() * 9)

```

**Q4 (a): Pascal**

```
procedure TestRandom(var Repetitions : integer);
var
    Frequency : array[1..10] : integer;
    Expected, NextRandom, N : integer;

begin
    Expected := Round(Repetitions/10);
    for N := 1 to 10 do
        Frequency[N] := 0;

    for N := 1 to Repetitions do
        begin
            NextRandom := random(10)+1;
            Frequency[NextRandom] := Frequency[NextRandom]+1;
        end;

        writeln ('The expected frequency is ', Expected);
        writeln ('Number Frequency Difference');

        for N := 1 to 10 do
            writeln ('    ', N, '    ', Frequency[N], '    ', Frequency[N] -
Expected);

        end;
```

**Q4 (a): Python**

```
# frequency is an array from 1 to 10 of type integer;
# nextNumber is an integer which stores the created random number
# expected is an integer which stores the expected frequency of each number

def TestRandom (repetitions):
    import random
    frequency = [0 for i in range(1,11)] # initialise each frequency count to 0

    expected = repetitions / 10

    for i in range(1, repetitions + 1):
        nextNumber = random.randint(1,10)
        frequency[nextNumber] = frequency[nextNumber]+ 1

    print ("The expected frequency is ", expected)
    print("    Number    Frequency    Difference")

    for i in range(1,11):
        print ("    ", i, "    ", frequency[i], "    ", frequency[i] -
expected)
```

Alternative:

```
def TestRandom (repetitions):
    expected = repetitions / 10      ## initialised as real/integer
                                   ## NextRandom and N defined as integers
    frequency =[0,0,0,0,0,0,0,0,0,0,0] ## defined as an array and
initialised to zero

    for n in range (0,repetitions):
        nextNumber = randint(1, 10)
        frequency[nextNumber] += 1

    print ('The expected frequency is ', expected)

    print ('Number    Frequency    Difference')
    for n in range (1, 11):
        print (n, '    ', frequency[n], '    ', frequency[n] - expected)
```

Alternative:

```
frequency =[0]*11      ## alternate way to initialise array to zero
frequency =[]          ## empty array/list
```

Alternative:

```
for n in range (1,11):
    frequency[n-1] = 0      ##alternate way to initialise array to zero
```

**Alternative:**

```
for n in range (0,11): ##alternate way to initialise array to 2  
    frequency.append(0)
```

QUESTION 9.

Cambridge
International
AS & A Level

Cambridge Assessment International Education
Cambridge International Advanced Subsidiary and Advanced Level



COMPUTER SCIENCE

9608/21

Paper 1 Written Paper

October/November 2018

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 International will not enter into discussions about these mark schemes.

Cambridge International is publishing the mark schemes for the October/November 2018 series for most Cambridge IGCSE™, Cambridge International A and AS Level components and some Cambridge O Level components.

This document consists of **12** printed pages.



Generic Marking Principles

These general marking principles must be applied by all examiners when marking candidate responses. They should be applied alongside the specific content of the mark scheme or generic level descriptors for a question. Each question paper and mark scheme will also comply with these marking principles.

GENERIC MARKING PRINCIPLE 1:

Marks must be awarded in line with:

- the specific content of the mark scheme or the generic level descriptors for the question
- the specific skills defined in the mark scheme or in the generic level descriptors for the question
- the standard of response required by a candidate as exemplified by the standardisation scripts.

GENERIC MARKING PRINCIPLE 2:

Marks awarded are always **whole marks** (not half marks, or other fractions).

GENERIC MARKING PRINCIPLE 3:

Marks must be awarded **positively**:

- marks are awarded for correct/valid answers, as defined in the mark scheme. However, credit is given for valid answers which go beyond the scope of the syllabus and mark scheme, referring to your Team Leader as appropriate
- marks are awarded when candidates clearly demonstrate what they know and can do
- marks are not deducted for errors
- marks are not deducted for omissions
- answers should only be judged on the quality of spelling, punctuation and grammar when these features are specifically assessed by the question as indicated by the mark scheme. The meaning, however, should be unambiguous.

GENERIC MARKING PRINCIPLE 4:

Rules must be applied consistently e.g. in situations where candidates have not followed instructions or in the application of generic level descriptors.

GENERIC MARKING PRINCIPLE 5:

Marks should be awarded using the full range of marks defined in the mark scheme for the question (however; the use of the full mark range may be limited according to the quality of the candidate responses seen).

GENERIC MARKING PRINCIPLE 6:

Marks awarded are based solely on the requirements as defined in the mark scheme. Marks should not be awarded with grade thresholds or grade descriptors in mind.



Question	Answer			
1(a)(i)	Statement	Selection	Repetition (Iteration)	Assignment
	WHILE Count < 20		✓	
	Count ← Count + 1			✓
	If MyGrade <> 'C' THEN	✓		
	Mark[Count] ← GetMark(StudentID)			✓
	ELSE OUTPUT "Fail"	✓		
	ENDFOR		✓	
	One mark for each row			
1(b)(i)	Statement		Data type	
	MyAverage ← 13.5		REAL	
	ProjectCompleted ← TRUE		BOOLEAN	
	Subject ← "Home Economics"		STRING	
	MyMark ← 270		INTEGER	
	MyGrade ← 'B'		CHAR	
1(b)(ii)	Expression		Evaluates to	
	"Air-" & MID(Subject, 7, 3)		"Air-con"	
	INT(MyAverage / 2)		6	
	ProjectCompleted AND MyMark > 270		FALSE	
	ProjectCompleted OR MyMark > 260		TRUE	
	ASC(MyGrade / 3)		ERROR	

5

5



Question	Answer	
2(a)	<pre> FUNCTION GetDiscountRate(CardNum : STRING) RETURNS REAL DECLARE DRate : REAL DECLARE Points : INTEGER DRate ← 0 Points ← GetPoints(CardNum) IF Points > 199 THEN DRate ← 0.2 ELSE IF Points > 99 THEN DRate ← 0.1 ENDIF ENDIF ENDIF IF Today() = 3 THEN DRate ← DRate * 1.2 ENDIF RETURN DRate ENDFUNCTION </pre> 1 mark for each of the following: 1 Correct FUNCTION heading (as given) and end 2 Declaring local variables for DRate and Points 3 Initialisation of DRate to zero and Points ← GetPoints(CardNum) 4 IF ... THEN ...(ELSE) ... ENDIF with Points > 199 5 (Nested) IF ... THEN ... ENDIF with Points > 99 6 ...correct assignments of DRate to 0.2 and 0.1 7 Checking Today() = 3 and increasing DRate by 20% 8 Return parameter // GetDiscountRate ← DRate Mark points 7 and 8 must not be nested	
2(b)(i)	Name: Syntax Description: Rules of programming language have not been followed Name: Logic Description: Where the program does not behave as expected / does not give the expected result / an error in the logic of the algorithm 1mark for name + 1 mark for corresponding description	2
2(b)(ii)	Name: Stub testing Description: A function could be written for GetPoints() that simply returns a test value or outputs a message (i.e. doesn't do the CardNum lookup)	2



Question	Answer	
2(c)(i)	1 mark for any of the following two values: 0.1 0.2 1.2 99 199 3	
2(c)(ii)	Example: CONSTANT MinDiscount = 0.1 1 mark for each of the following: • meaningful identifier name and corresponding value • correct syntax	2
2(c)(iii)	1 mark for: • The value cannot accidentally get changed // be different in two places • A change to the value requires changing in one place only / don't have to repeatedly write out the same value throughout the program	2
2(c)(iv)	Tried and tested // pre compiled (contains no syntax errors)	1
2(c)(v)	1 mark for feature (Name) and 1 mark for corresponding description (explanation) Example: Name: Meaningful variable names Explanation: To reduce the risk of referring to the wrong variable / make the code easier to understand Name: Indentation Explanation: To see where loops / selection start / end // indicate program structure Name: Variable type-checking as part of module interface Explanation: Reduces the risk of using an incorrect parameter Name: Pretty-Printing Explanation: Highlights the error / auto-complete / type checking Name: / <u>Dynamic</u> Syntax Checking Explanation: Highlights the error as code is typed in	2



Question	Answer																	
3(a)	Code has to be in machine code (or equivalent) to be executed																	
3(b)	1 mark for the name (what you do) and one for description (how) For example: Method: • Dry run the code // use of white box testing // trace tables • Trace the contents of variables // trace all possible routes through the program Method: • Breakpoints • Run the code to a set point to find error Method: • Variable watch • Check the contents of variables at specific points in the program Method: • Stepping • Execute the code line by line Method: • Include OUTPUT statements in the code • to display the value of variables as the code was running																	
3(c)	<table><tr><th>Statement</th><th>White-box</th><th>Black-box</th></tr><tr><td>The student does not need to know the structure of the code.</td><td></td><td>✓</td></tr><tr><td>The student chooses data to test every possible path through the code.</td><td>✓</td><td></td></tr><tr><td>The student chooses normal, boundary and erroneous data.</td><td>✓</td><td>(✓)</td></tr><tr><td>The student chooses data to test that the program meets the specification.</td><td></td><td>✓</td></tr></table> 1 mark per row			Statement	White-box	Black-box	The student does not need to know the structure of the code.		✓	The student chooses data to test every possible path through the code.	✓		The student chooses normal, boundary and erroneous data.	✓	(✓)	The student chooses data to test that the program meets the specification.		✓
Statement	White-box	Black-box																
The student does not need to know the structure of the code.		✓																
The student chooses data to test every possible path through the code.	✓																	
The student chooses normal, boundary and erroneous data.	✓	(✓)																
The student chooses data to test that the program meets the specification.		✓																

4

4



Question	Answer	
4(a)(i)	The identifier name of a global integer referenced	NumElements
	The identifier name of a user-defined procedure	SaveToFile
	The line number of an unnecessary statement	16
	The scope of <code>ArrayString</code>	Local
4(a)(ii)	1 mark for each mark point: - Loop / repeat / iterate through array <code>ResultArray</code> one element at a time - extract a string from <u>row / column 1</u> of the array - compare the string with <code>SearchString</code> - if they match, call <code>SaveToFile()</code> and increment <code>NumberFound</code>	4
4(b)	Pseudocode solution included here for development and clarification of mark scheme. Programming language solutions appear at the end of this mark scheme. <pre> FUNCTION ScanArray(SearchString : STRING) RETURNS INTEGER DECLARE ArrayIndex : INTEGER DECLARE ArrayString : STRING DECLARE NumberFound : INTEGER NumberFound ← 0 FOR ArrayIndex ← 1 TO NumElements ArrayString ← ResultArray[ArrayIndex, 1] IF TO_UPPER(ArrayString) = TO_UPPER(SearchString) THEN CALL SaveToFile(ArrayString) NumberFound ← NumberFound + 1 ENDIF ENDFOR RETURN NumberFound ENDFUNCTION </pre> 1 mark for each of the following: - Function header and end including parameter and return - Declaration of two local variables as above but NOT <code>NumElements</code> - FOR ... ENDFOR loop with range as given - Referencing each element from the array - Converting both strings to uppercase / lowercase - If strings are equal then Call <code>SaveToFile()</code> and increment <code>NumberFound</code>	6



Question	Answer	
4(c)	1 mark for name; 1 mark for each advantage (max 2) Name: Stepwise refinement // Top-down design // Modularisation // Decomposition Advantage: • Makes the problem / task / algorithm easier to understand // reduce program complexity • Smaller modules easier to develop / test / debug • Programmers can work on different modules // different expertise	
4(d)	Pseudocode solution included here for development and clarification of mark scheme. Programming language solutions appear at the end of this mark scheme. <pre> DECLARE ResultArray : ARRAY [1:100, 1:2] OF STRING DECLARE i, j : INTEGER FOR i ← 1 to 100 FOR j ← 1 to 2 ResultArray[i, j] ← '*' ENDFOR ENDFOR </pre> One mark for: • <code>ResultArray</code> declaration / commented in Python • assigning to all elements • assignment of <code>'*'</code>	3



Question	Answer
5	<pre> FUNCTION SaveStatus() RETURNS BOOLEAN DECLARE Time : STRING DECLARE Fuel : STRING DECLARE Distance : STRING DECLARE FileData : STRING DECLARE Tries : INTEGER DECLARE ReturnFlag : BOOLEAN Tries ← 1 ReturnFlag ← TRUE Distance ← GetDistance() Fuel ← GetFuel() Time ← GetTime() WHILE Time = NULL AND Tries < 3 Time ← GetTime() Tries ← Tries + 1 ENDWHILE IF Time = NULL THEN ReturnFlag ← FALSE ELSE FileData ← Time & ',' & Fuel & ',' & Distance OPENFILE "CarStatus.txt" FOR APPEND WRITEFILE "CarStatus.txt", FileData CLOSEFILE "CarStatus.txt" ENDIF RETURN ReturnFlag ENDFUNCTION </pre> 1 mark for each of the following: 1 Function heading as shown 2 Declare Time local variable as STRING 3 Calls GetDistance() and GetFuel() once 4 Loop (up to three times or) until Time <> NULL 5 Call GetTime() in a loop 6 Return FALSE if 3 NULLs 7 Open file in APPEND mode 8 Forming the text string with comma separators and write to the file 9 OPEN ... WRITE ... CLOSE as three lines not separated by loop 10 Return TRUE



Program Code Solutions

Q4 (b): Visual Basic

```
Function ScanArray(SearchString As String) As Integer

    Dim ArrayIndex As Integer
    Dim ArrayString As String
    Dim NumberFound As Integer

    NumberFound = 0

    For ArrayIndex = 1 To NumElements
        ArrayString = ResultArray(ArrayIndex, 1)
        If UCase(ArrayString) = UCase(SearchString) Then
            Call SaveToFile(ArrayString)
            NumberFound = NumberFound + 1
        End If
    Next ArrayIndex
    Return NumberFound

End Function
```

Q4 (b): Pascal

```
function ScanArray(SearchString : String) : Integer;

var
    ArrayIndex : Integer;
    ArrayString : String;
    NumberFound : Integer;

begin
    NumberFound := 0;

    For ArrayIndex := 1 To NumElements do
        begin
            ArrayString := ResultArray[ArrayIndex, 1];
            If ToUpper(ArrayString) = ToUpper(SearchString) then
                begin
                    SaveToFile(ArrayString); // Keyword "Call" not valid
                    NumberFound := NumberFound + 1;
                end;
            end;
        end;

    Result := NumberFound; // ScanArray := NumberFound
end.
```


**Q4 (b): Python**

```
def ScanArray(SearchString):  
  
    # ArrayIndex : integer  
    # ArrayString : string  
    # NumberFound : integer  
  
    NumberFound = 0  
  
    for ArrayIndex in range(NumElements): # 0 to NumElements-1  
        ArrayString = ResultArray[ArrayIndex][0]  
        if ArrayString.upper == SearchString.upper:  
            SaveToFile(ArrayString) # Keyword "Call" not valid  
            NumberFound = NumberFound + 1  
  
    return NumberFound # ScanArray := NumberFound
```

**Q4 (d): Visual Basic**

```
Dim ResultArray(100, 2) As String
Dim i, j As Integer

For i = 1 to 100
    For j = 1 to 2
        ResultArray(i, j) = '*'
    Next j
Next i
```

Q4 (d): Pascal

```
var
    ResultArray : array[1..100, 1..2] of string;
    i, j : integer;
begin
    For i := 1 to 100 do
        For j := 1 to 2 do
            begin
                ResultArray[i, j] := '*';
            end;
        end;
    end.
```

Q4 (d): Python

```
# ResultArray[1..100, 1..2] : String

ResultArray = [[''] * 2 for i in range(100)]

for i in range(100):
    for j in range(2):
        ResultArray[i][j] = '*'
```

Q4 (d): Python – alternative 1 of n

```
# ResultArray[1..100, 1..2] : String

ResultArray = [['*'] * 2 for i in range(100)]
```

Q4 (d): Python – alternative 2 of n

```
# ResultArray[1..100, 1..2] : String

ResultArray = [['*'] * 2] * 100
```

QUESTION 10.



Question	Answer	
3(a)	You should be able to recognise / understand in another language: Declaration / assignment / sequence / selection / repetition (iteration) / Subroutines / Parameters passed between modules / program structure / Input and Output Any 2 marks from the list. Max 1 mark if no explanation given	
3(b)	Key points: - to increase the level of detail of the algorithm // break the problem into smaller steps - until steps are easier to solve // to be directly translated into lines of code	2
3(c)	Key points: - A <u>loop / repetition / iteration</u> to check every element - Compare the array element with the value being searched - Exit loop / stop search when value found or end of array reached - If value is found then output the index position, otherwise output "Not found"	4

Question	Answer	Marks

QUESTION 11.



Question	Answer
3(a)	<pre> TotalValue ← 0 ZeroCount ← 0 FOR Index ← 1 TO 100 TotalValue ← TotalValue + Result[Index] IF Result[Index] = 0.0 THEN ZeroCount ← ZeroCount + 1 ENDIF ENDFOR OUTPUT "The average is ", (TotalValue / 100) OUTPUT "The number of elements with a zero value is ", ZeroCount One mark for each of the following: 1 Both initialisations 2 Loop 100 times 3 Adding individual element to TotalValue in a loop 4 Check if element value is zero in a loop 5 If so increment ZeroCount in a loop 6 Average is calculated after the loop 7 Both OUTPUT statements, including message and variables </pre>
3(b)	<u>PROCEDURE ScanArray</u> (<u>BYREF AverageValue: REAL</u>, <u>BYREF ZeroCount: INTEGER</u>, <u>ArrayName : ARRAY</u>) One mark for each underlined part Names unimportant but first two parameters must be BYREF

QUESTION 12.

Cambridge
International
AS & A Level

Cambridge Assessment International Education
Cambridge International Advanced Subsidiary and Advanced Level



COMPUTER SCIENCE

9608/22

Paper 2 Written Paper

May/June 2019

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 International will not enter into discussions about these mark schemes.

Cambridge International is publishing the mark schemes for the May/June 2019 series for most Cambridge IGCSE™, Cambridge International A and AS Level and Cambridge Pre-U components, and some Cambridge O Level components.

This document consists of **15** printed pages.



Cambridge Assessment
International Education



Generic Marking Principles

These general marking principles must be applied by all examiners when marking candidate responses. They should be applied alongside the specific content of the mark scheme or generic level descriptors for a question. Each question paper and mark scheme will also comply with these marking principles.

GENERIC MARKING PRINCIPLE 1:

Marks must be awarded in line with:

- the specific content of the mark scheme or the generic level descriptors for the question
- the specific skills defined in the mark scheme or in the generic level descriptors for the question
- the standard of response required by a candidate as exemplified by the standardisation scripts.

GENERIC MARKING PRINCIPLE 2:

Marks awarded are always **whole marks** (not half marks, or other fractions).

GENERIC MARKING PRINCIPLE 3:

Marks must be awarded **positively**:

- marks are awarded for correct/valid answers, as defined in the mark scheme. However, credit is given for valid answers which go beyond the scope of the syllabus and mark scheme, referring to your Team Leader as appropriate
- marks are awarded when candidates clearly demonstrate what they know and can do
- marks are not deducted for errors
- marks are not deducted for omissions
- answers should only be judged on the quality of spelling, punctuation and grammar when these features are specifically assessed by the question as indicated by the mark scheme. The meaning, however, should be unambiguous.

GENERIC MARKING PRINCIPLE 4:

Rules must be applied consistently e.g. in situations where candidates have not followed instructions or in the application of generic level descriptors.

GENERIC MARKING PRINCIPLE 5:

Marks should be awarded using the full range of marks defined in the mark scheme for the question (however; the use of the full mark range may be limited according to the quality of the candidate responses seen).

GENERIC MARKING PRINCIPLE 6:

Marks awarded are based solely on the requirements as defined in the mark scheme. Marks should not be awarded with grade thresholds or grade descriptors in mind.



Question	Answer													
1(a)(i)	- a sequence of steps / stages / instructions - to implement a task // solution to a problem Allow alternatives to sequence providing meaning is clear.													
1(a)(ii)	Input: - e.g. <code>INPUT MyVar // READFILE MyFile, MyString</code> Process: - e.g. <code>NextChar ← 'X' // Count ← Count + 1</code> Output:: - e.g. <code>OUTPUT "Hello World" // WRITEFILE "LogFile.txt", SomeData</code> Mark as follows: One mark for each stage (Input and Process) One mark for each pseudocode example													
1(b)(i)	<table><tr><th>Expression</th><th>Evaluates to</th></tr><tr><td><code>STRING_TO_NUM(RIGHT(ID, 3))</code></td><td>234.0 / 234</td></tr><tr><td><code>INT(Height * Children)</code></td><td>11</td></tr><tr><td><code>IsMarried AND Married < 31/12/1999</code></td><td>TRUE</td></tr><tr><td><code>LENGTH(ID & NUM_TO_STRING(Height))</code></td><td>8</td></tr><tr><td><code>MID((ID, INT(Height) - Children, 2)</code></td><td>"23"</td></tr></table> No quotes for row 1 Quotes (single or double) for row 5	Expression	Evaluates to	<code>STRING_TO_NUM(RIGHT(ID, 3))</code>	234.0 / 234	<code>INT(Height * Children)</code>	11	<code>IsMarried AND Married < 31/12/1999</code>	TRUE	<code>LENGTH(ID & NUM_TO_STRING(Height))</code>	8	<code>MID((ID, INT(Height) - Children, 2)</code>	"23"	5
Expression	Evaluates to													
<code>STRING_TO_NUM(RIGHT(ID, 3))</code>	234.0 / 234													
<code>INT(Height * Children)</code>	11													
<code>IsMarried AND Married < 31/12/1999</code>	TRUE													
<code>LENGTH(ID & NUM_TO_STRING(Height))</code>	8													
<code>MID((ID, INT(Height) - Children, 2)</code>	"23"													
1(b)(ii)	<table><tr><th>Variable</th><th>Data type</th></tr><tr><td><code>Married</code></td><td>DATE</td></tr><tr><td><code>ID</code></td><td>STRING</td></tr><tr><td><code>MiddleInitial</code></td><td>CHAR</td></tr><tr><td><code>Height</code></td><td>REAL</td></tr><tr><td><code>IsMarried</code></td><td>BOOLEAN</td></tr></table> One mark per data type	Variable	Data type	<code>Married</code>	DATE	<code>ID</code>	STRING	<code>MiddleInitial</code>	CHAR	<code>Height</code>	REAL	<code>IsMarried</code>	BOOLEAN	5
Variable	Data type													
<code>Married</code>	DATE													
<code>ID</code>	STRING													
<code>MiddleInitial</code>	CHAR													
<code>Height</code>	REAL													
<code>IsMarried</code>	BOOLEAN													



Question	Answer	
2(a)(i)	- To make a more manageable / understandable solution - To support modular design	
2(a)(ii)	- Allows the subroutine to be called from many / multiple places - Subroutine may be (independently) tested and debugged - If the task changes the change needs to be made only once - Reduces unnecessary duplication / program lines - Allows teams to work on different parts of the solution	
2(a)(iii)	Type of subroutine: Function Justification: It returns a value // assigns a value to variable <i>Answer</i> One mark for type One mark for justification	2
2(b)	- An editor is used to produce / write / modify the <u>source code</u> / <u>program</u> / <u>high-level language code</u> OR by example: An editor provides (features such as) context-sensitive prompts / dynamic syntax checking etc. - A translator (compiler) is used to translate / convert the source code / program / high-level language code into <u>object code</u> / <u>machine code</u> / <u>an executable file</u>. OR A translator (interpreter) is used to translate the source code / program / high-level language code <u>line by line</u> - A debugger is used to test the program / detect errors (and correct errors) in the program. One mark per bullet point	3



Question	Answer
2(c)	Control structure: A (pre-) <u>conditional</u> loop Function of code: • Check if <code>Result</code> is less than 20 and If true, calls <code>ResetSensor</code> with parameter value 3... • ... and assign the value returned by <code>GetSensor</code> with parameter value 3 to <code>Result</code> • Loop until <code>Result >= 20</code> OR Control structure: A selection // conditional statement Function of code: • Check if <code>Result</code> is less than 20 and If true, calls <code>ResetSensor</code> with parameter value 3... • ... and assign the value returned by <code>GetSensor</code> with parameter value 3 to <code>Result</code> One mark for control structure, maximum two for function Function of code marks independent of answer to control structure

Question	Answer	Marks
3(a)(i)	<u>PROCEDURE SubA</u> (<u>A : STRING</u>, <u>B : INTEGER</u>, <u>BYREF C : CHAR</u>) One mark for each underlined part Ignore <code>BYVAL</code> for parameter A and/or parameter B Parameter order / names not important but must be correct data types	3
3(a)(ii)	<u>Function SubB</u> (<u>D : STRING</u>, <u>E : INTEGER</u>) <u>RETURNS BOOLEAN</u> One mark for each underlined part Ignore <code>BYVAL</code> for parameter D and/or parameter E Parameter order / names not important but must be correct data types	3
3(b)	• Selection • Iteration • Sequence One mark per bullet to max. 2	2



Question	Answer				
4(a)(i)	Index	NextChar	Selected	NewValue	NewString
			0		"0"
	1	'1'			"01"
	2	'2'			"012"
	3	'▽'		12	
			12		
					"0"
	4	'3'			"03"
	5	'4'			"034"
	6	'▽'		34	
			34		
					"0"
	7	'5'			"05"
	8	'▽'		5	
					"0"
	9	'▽'		0	
					"0"
	10	'3'			"03"
	11	'9'			"039"
One mark for each column.					
If no mark for columns, award one mark for initialisation of <code>Selected</code> to 0 and <code>Newstring</code> to '0' (single or double quotes).					
4(a)(ii)	34				



Question	Answer	
4(b)(i)	- The final value (in the string) is the largest value (39) and is not considered // final comparison with variable <code>Selected</code> is not made - The loop terminates at the end of the string (the character 9) // there wasn't a final space / non-numeric digit One mark per bullet.	
4(b)(ii)	- Check the (final) value of <code>NewString</code> after the loop... ...and see if it is greater than <code>Selected</code> (repeat the existing conditional clause) OR - Amend the algorithm to add a space character / non-numeric character to the end of the string... ...before the <code>FOR</code> loop / at the start of the function One mark per bullet point Accept alternative workable solution	2



Question	Answer	
5(a)	One mark for each of: • Open the file • Set a count to zero • Loop until end of file // no more lines to read • Increment the count each time a line is read in a loop Maximum 3 marks	
5(b)	<pre>PROCEDURE CountLines(FileName : STRING) DECLARE NumLines : INTEGER DECLARE Dummy : STRING NumLines ← 0 OPENFILE FileName FOR READ WHILE NOT EOF(FileName) READFILE FileName, Dummy NumLines ← NumLines + 1 ENDWHILE CLOSEFILE FileName OUTPUT "Number of lines in the file : ", NumLines ENDPROCEDURE</pre> One mark for each of the following: 1. Procedure header and end, including parameter 2. Declaration and initialisation of a local INTEGER to count lines (e.g. NumLines) 3. OPEN file in read mode and CLOSE file 4. WHILE loop stopping when EOF(FileName) 5. Read a line from the file and increment NumLines in a loop 6. Output a message plus the NumLines outside a loop	6



Question	Answer
6(a)	'Pseudocode' solution included here for development and clarification of mark scheme. Programming language example solutions appear in the Appendix. <pre> FUNCTION GetInfo() RETURNS STRING DECLARE ID : STRING DECLARE PreferredName : STRING DECLARE Valid : BOOLEAN Valid ← FALSE WHILE Valid = FALSE OUTPUT "Please Enter a valid ID" INPUT ID IF LENGTH(ID) = 5 AND LEFT(ID, 1) >= 'A' <u> </u> AND LEFT(ID, 1) <= 'Z' <u> </u> AND ISNUM(RIGHT(ID, 4)) <u> </u> THEN Valid ← TRUE ENDIF ENDWHILE OUTPUT "Please enter preferred name" INPUT PreferredName RETURN ID & '*' & PreferredName ENDFUNCTION </pre> One mark for each of the following: 1. Function header and end (where appropriate) 2. Local variables used are declared (commented in python) 3. Prompt and input for ID (until valid) and preferred name 4. Conditional loop repeating while ID is invalid 5. test length in a loop 6. test first character in a loop 7. test last four characters in a loop 8. Concatenate using correct separator character and return resulting string



Question	Answer	
6(b)	'Pseudocode' solution included here for development and clarification of mark scheme. Programming language example solutions appear in the Appendix. <pre> PROCEDURE TopLevel() DECLARE Response : CHAR DECLARE InputData : STRING DECLARE Success : BOOLEAN Response ← 'Y' WHILE Response = 'Y' InputData ← GetInfo() IF LEFT(InputData,1) < 'N' THEN Success ← WriteInfo(InputData, "File1.txt") ELSE Success ← WriteInfo(InputData, "File2.txt") ENDIF IF NOT Success THEN Response ← 'N' ELSE OUTPUT "Enter details for another student? (Y/N)" INPUT Response ENDIF ENDWHILE ENDPROCEDURE </pre> One mark for each of the following: 1. Procedure header and end 2. Conditional loop terminated with user input 3. call to <code>GetInfo()</code> in a loop 4. check first character of returned <code>UserID</code> value in a loop 5. call(s) to <code>WriteInfo()</code> in both cases ... 6. ... with two <code>STRING</code> parameters in a loop 7. exit procedure if <code>WriteInfo()</code> unsuccessful in a loop 8. if <code>WriteInfo()</code> successful, prompt and check input to repeat / exit in a loop	
6(c)	<u>FUNCTION WriteInfo (FileData : STRING, Filename : STRING)</u> <u>RETURNS BOOLEAN</u> One mark per underlined section	3

**Program Code Example Solutions****Q6 (a): Visual Basic**

```
Function GetInfo() As String
    Dim ID As String = ""
    Dim PreferredName As String = ""
    Dim Valid As Boolean = False
    While Valid = False
        Console.WriteLine("Please enter a valid ID : ")
        ID = Console.ReadLine()
        If Len(ID) = 5 And Left(ID, 1) >= "A" And Left(ID, 1) <= "Z" ____
            And IsNumeric(Right(ID, 4)) Then
                Valid = True
            End If
        End While
        Console.WriteLine("Please enter preferred name : ")
        PreferredName = Console.ReadLine()
        Return ID & "*" & PreferredName
    End Function
```

Alternative:

```
Function GetInfo() As String

    Dim ID As String
    Dim PreferredName As String
    Dim Valid As Boolean
    Dim Number As String
    Dim Size As Integer
    Dim i As Integer

    Valid = False

    While Valid = False
        Console.WriteLine("Please Enter a valid ID")
        ID = Console.ReadLine()
        Size = Len(ID)

        If (Size = 5) And ((Left(ID, 1) >= "A") And (Left(ID, 1) <= "Z"))
Then
            Valid = True
            For i = 2 To 5
                Number = Mid(ID, i, 1)
                If (Number < "0") Or (Number > "9") Then
                    Valid = False
                End If
            Next
        End If
    End While

    Console.WriteLine("Please enter preferred name")
    PreferredName = Console.ReadLine()
    Return (ID & "*" & PreferredName)
End Function
```

**Q6 (a): Pascal**

```
function GetInfo() : String;
var
    ID : String;
    PreferredName : String;
    Valid : Boolean;
    Value, Code : Integer;
begin
    Valid := false;
    while not Valid do
    begin
        Write('Please enter a valid ID : ');
        Readln(ID);
        if (Length(ID) = 5) and (ID[1] >= 'A') and (ID[1] <= 'Z') then
            Valid := true;
        Val(Copy(ID, 2, 4), Value, Code);
        if Code <> 0 then
            Valid := false;
        end;
        Write('Please enter preferred name : ');
        Readln(PreferredName);
        GetInfo := ID + '*' + PreferredName;
    end;
end;
```

Free Pascal

```
function GetInfo() : String;
var
    ID : String;
    PreferredName : String;
    Valid : Boolean;
    Value, Code : Integer;
begin
    Valid := false;
    while not Valid do
    begin
        Write('Please enter a valid ID : ');
        Readln(ID);
        if (Length(ID) = 5) and (ID[1] >= 'A') and (ID[1] <= 'Z') and (IsNumber(SubStr(ID, 2, 4))) then
            Valid := true;
        end;
        Write('Please enter preferred name : ');
        Readln(PreferredName);
        result := ID + '*' + PreferredName;
    end;
end;
```


**Q6 (a): Python**

```
def GetInfo() :  
    ID = ""                # string variable  
    PreferredName = ""     # string variable  
    Valid = False         # Boolean variable  
    while not Valid :  
        ID = input("Please enter a valid ID : ")  
        if len(ID) == 5 and ID[0] >= "A" and ID[0] <= "Z" and  
ID[1:].isnumeric() :  
            Valid = True  
    PreferredName = input("Please enter preferred name : ")  
    return ID + "*" + PreferredName
```

**Q6 (b): Visual Basic**

```
Sub TopLevel()  
    Dim Response As String = "Y"  
    Dim InputData As String = ""  
    Dim Success As Boolean = True  
    While Response = "Y"  
        InputData = GetInfo()  
        If Left(InputData, 1) < "N" Then  
            Success = WriteInfo(InputData, "File1.txt")  
        Else  
            Success = WriteInfo(InputData, "file2.txt")  
        End If  
        If Not Success Then  
            Response = "N"  
        Else  
            Console.Write("Enter details for another student? Y/N ")  
            Response = Console.ReadLine()  
        End If  
    End While  
End Sub
```

Q6 (b): Pascal

```
procedure TopLevel();  
var  
    Response : Char;  
    InputData : String;  
    Success : Boolean;  
begin  
    Response := 'Y';  
    while Response = 'Y' do  
        begin  
            InputData := GetInfo();  
            if InputData[1] < 'N' then  
                Success := WriteInfo(InputData, 'File1.txt')  
            else  
                Success := WriteInfo(InputData, 'File2.txt');  
            if not Success then  
                Response := 'N'  
            else  
                begin  
                    Write('Enter details for another student? (Y/N) ');  
                    Readln(Response);  
                end;  
            end;  
        end;  
    end;  
end;
```

**Q6 (b): Python**

```
def TopLevel() :  
    Response = "Y"           # string/character variable  
    InputData = ""           # string variable  
    Success = True           # Boolean variable  
    while Response == "Y" :  
        InputData = GetInfo()  
        if InputData[0] < "N" :  
            Success = WriteInfo(InputData, "File1.txt")  
        else :  
            Success = WriteInfo(InputData, "File2.txt")  
        if not Success :  
            Response = "Y"  
        else :  
            Response = input("Enter details for another student? (Y/N) ")
```

QUESTION 13.



Question	Answer	Marks
5(a)(i)	To test <u>every path</u> through the code / algorithm Accept phrase with equivalent meaning	1
5(a)(ii)	A trace table	1
5(b)(i)	String: - three possible formats for string containing two words: "Cat∇∇Dog" // "Cat∇Dog∇" // "∇Cat∇Dog" Explanation: - When a space character is encountered - NumWords is incremented by 1 OR: - The algorithm counts the spaces - and not the words 1 mark for a string that would give the correct result 2 marks for explanation	3



Question	Answer
5(b)(ii)	String 1: • Output: <code>Number of words : 2</code> Description 1: • Check character at end of string • If not a character, increment variable <code>NumTotal</code> OR: • Add a space at the beginning / end of string • At the start of the algorithm OR: • Change Initialisation of <code>NumWords</code> • to 1 OR: • After the loop • Add 1 to <code>NumWords</code> String 2: • Output: <code>Number of words : 5</code> Description 2: • Detect a space followed by a space • Count as a single space / only increment variable <code>NumTotal</code> once OR: • Replace all double spaces with a single space • Before the loop OR: • After the loop • Subtract 2 from <code>NumWords</code> Many possible solutions. One mark for each correct output One marks for each description bullet point

QUESTION 14.



Question	Answer
4(a)	One mark for process name, max 3 for structured English. Process: - Stepwise Refinement / Top-down design Structured English: - Check that character is between 'A' and 'Z' - Produce unique array index for this character - Increment this array element
4(b)	<pre> DECLARE Index : INTEGER DECLARE Count : INTEGER FOR Count ← 1 TO LENGTH(InString) NextChar ← UCASE(MID(InString, Count, 1)) IF NextChar >= 'A' AND NextChar <= 'Z' THEN Index ← ASC(NextChar) - 64 Result[Index] ← Result[Index] + 1 ENDIF ENDFOR FOR Index ← 1 TO 26 OUTPUT "Letter " & CHR(Index + 64) & " : " & NUM_TO_STRING(Result[Index]) ENDFOR </pre> One mark for each of the following (max 7): 1 First loop from 1 to length of InString: 2 Extract each character in turn in a loop 3 Check that character is alphabetic (must cater for lower & upper case) in a loop 4 Obtain array index using ASC() - 64 in a loop 5 Increment element of Result array in a loop 6 Second loop from 1 to 26: 7 Attempt to OUTPUT character A to Z and corresponding count in a loop 8 Fully complete OUTPUT including any necessary type conversion in a loop

7

QUESTION 15.



Question	Answer	
1(a)(i)	Any three from: 1. Indentation 2. Blank lines / white space 3. Sensible identifier names / use of Camel Case for identifier names 4. Capitalised keywords	
1(a)(ii)	START Tries ← 1 Full ← ReadSensor("F1") Is Full = TRUE? Is NOT Full AND Tries < 4 Call TopUp() ReadSensor("F1") Tries ← Tries + 1 Output "Already Full" Is Tries > 3? Output "Too many attempts" Output "Tank now full" Mark as follows: • One mark for START and END • One mark for each of the 4 areas outlined Loop must be included for this area mark	5



Question	Answer											
1(b)(i)	One mark per row <table><tr><th>Example value</th><th>Data type</th></tr><tr><td>43</td><td>INTEGER</td></tr><tr><td>TRUE</td><td>BOOLEAN</td></tr><tr><td>-273.16</td><td>REAL</td></tr><tr><td>"-273.16"</td><td>STRING</td></tr></table>	Example value	Data type	43	INTEGER	TRUE	BOOLEAN	-273.16	REAL	"-273.16"	STRING	
Example value	Data type											
43	INTEGER											
TRUE	BOOLEAN											
-273.16	REAL											
"-273.16"	STRING											
1(b)(ii)	One mark per row <table><tr><th>Expression</th><th>Evaluates to</th></tr><tr><td>RIGHT("Stop", 3) & LEFT("ich",2)</td><td>"topic"</td></tr><tr><td>MID(NUM_TO_STRING(2019), 3, 1)</td><td>"1"</td></tr><tr><td>INT(NUM_TO_STRING(-273.16))</td><td>ERROR</td></tr><tr><td>INT(13/2)</td><td>6</td></tr></table>	Expression	Evaluates to	RIGHT("Stop", 3) & LEFT("ich",2)	"topic"	MID(NUM_TO_STRING(2019), 3, 1)	"1"	INT(NUM_TO_STRING(-273.16))	ERROR	INT(13/2)	6	4
Expression	Evaluates to											
RIGHT("Stop", 3) & LEFT("ich",2)	"topic"											
MID(NUM_TO_STRING(2019), 3, 1)	"1"											
INT(NUM_TO_STRING(-273.16))	ERROR											
INT(13/2)	6											

QUESTION 16.



Question	Answer	Marks
4(a)	One mark for each point. Valid string must contain: • at least one '.' characters • one '@' character • more than 5 other characters.	3



Question	Answer																																																																																																														
4(b)(i)	One mark for each area as outlined: <table><thead><tr><th>Index</th><th>NextChar</th><th>NumDots</th><th>NumAts</th><th>NumOthers</th></tr></thead><tbody><tr><td></td><td></td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>'J'</td><td></td><td></td><td>1</td></tr><tr><td>2</td><td>'i'</td><td></td><td></td><td>2</td></tr><tr><td>3</td><td>'m'</td><td></td><td></td><td>3</td></tr><tr><td>4</td><td>'.'</td><td>1</td><td></td><td></td></tr><tr><td>5</td><td>'9'</td><td></td><td></td><td>4</td></tr><tr><td>6</td><td>'9'</td><td></td><td></td><td>5</td></tr><tr><td>7</td><td>'@'</td><td></td><td>1</td><td></td></tr><tr><td>8</td><td>'s'</td><td></td><td></td><td>6</td></tr><tr><td>9</td><td>'k'</td><td></td><td></td><td>7</td></tr><tr><td>10</td><td>'a'</td><td></td><td></td><td>8</td></tr><tr><td>11</td><td>'i'</td><td></td><td></td><td>9</td></tr><tr><td>12</td><td>'l'</td><td></td><td></td><td>10</td></tr><tr><td>13</td><td>'.'</td><td>2</td><td></td><td></td></tr><tr><td>14</td><td>'c'</td><td></td><td></td><td>11</td></tr><tr><td>15</td><td>'o'</td><td></td><td></td><td>12</td></tr><tr><td>16</td><td>'m'</td><td></td><td></td><td>13</td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr></tbody></table>	Index	NextChar	NumDots	NumAts	NumOthers			0	0	0	1	'J'			1	2	'i'			2	3	'm'			3	4	'.'	1			5	'9'			4	6	'9'			5	7	'@'		1		8	's'			6	9	'k'			7	10	'a'			8	11	'i'			9	12	'l'			10	13	'.'	2			14	'c'			11	15	'o'			12	16	'm'			13																				
Index	NextChar	NumDots	NumAts	NumOthers																																																																																																											
		0	0	0																																																																																																											
1	'J'			1																																																																																																											
2	'i'			2																																																																																																											
3	'm'			3																																																																																																											
4	'.'	1																																																																																																													
5	'9'			4																																																																																																											
6	'9'			5																																																																																																											
7	'@'		1																																																																																																												
8	's'			6																																																																																																											
9	'k'			7																																																																																																											
10	'a'			8																																																																																																											
11	'i'			9																																																																																																											
12	'l'			10																																																																																																											
13	'.'	2																																																																																																													
14	'c'			11																																																																																																											
15	'o'			12																																																																																																											
16	'm'			13																																																																																																											
4(b)(ii)	TRUE																																																																																																														



Question	Answer
4(c)	One mark for string and one mark for correct explanation. Same for second answer providing it results in a different path through the algorithm. Correct answers may be: • without the correct number of '.' • without the correct number of '@' • without the correct number of 'other characters'