

QUESTION 1.



- 5 (a) • 1D Array // List
- INTEGER

[1]

[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 ' <u>found</u> '	(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.WriteLine(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 2.



- 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]

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



- (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 3.



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 4.



6 (a) (i) 12 [1]

(ii) 'L' [1]
Note: quotes are optional – must be upper case L

(b) (i)

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

[1]

[1]

[1]

[1]

Note: Correct (identifier + the data type + description) needed to score

Page 8	Mark Scheme
	Cambridge International AS/A Level – October/November 2015



```

(ii) // main program
      INPUT MyString
      ChangedString ← RemoveSpaces(MyString)
      OUTPUT ChangedString

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

```

1

1

1

1

1

1

1

1

1 only awarded if follows the previous mark

[Max 7]

QUESTION 5.



6 (a)	<table><tr><th>n</th><th>f</th><th>x</th><th>y</th><th>MID(String1, x, 1)</th><th>MID(String2, y, 1)</th></tr><tr><td>0</td><td>0</td><td></td><td></td><td></td></tr><tr><td>1</td><td></td><td>1</td><td>1</td><td>'R'</td><td>'R'</td></tr><tr><td></td><td></td><td>2</td><td>2</td><td>'E'</td><td>'A'</td></tr><tr><td>2</td><td></td><td>2</td><td>1</td><td>'E'</td><td>'R'</td></tr><tr><td>3</td><td></td><td>3</td><td>1</td><td>'T'</td><td>'R'</td></tr><tr><td>4</td><td></td><td>4</td><td>1</td><td>'R'</td><td>'R'</td></tr><tr><td></td><td></td><td>5</td><td>2</td><td>'A'</td><td>'A'</td></tr><tr><td></td><td></td><td>6</td><td>3</td><td>'C'</td><td>'C'</td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td></tr></table> One mark per correct column - Minus one mark if anything else on first row - Column 2 – '4' must not precede '4' in column 1 (as shown by arrow) - Letters must all be in upper case - Ignore quotation symbol	n	f	x	y	MID(String1, x, 1)	MID(String2, y, 1)	0	0				1		1	1	'R'	'R'			2	2	'E'	'A'	2		2	1	'E'	'R'	3		3	1	'T'	'R'	4		4	1	'R'	'R'			5	2	'A'	'A'			6	3	'C'	'C'							
	n	f	x	y	MID(String1, x, 1)	MID(String2, y, 1)																																																							
	0	0																																																											
	1		1	1	'R'	'R'																																																							
			2	2	'E'	'A'																																																							
	2		2	1	'E'	'R'																																																							
	3		3	1	'T'	'R'																																																							
	4		4	1	'R'	'R'																																																							
			5	2	'A'	'A'																																																							
			6	3	'C'	'C'																																																							
(b) (i)	- to search for a string within another string / String2 within String1 - to return the position of the start of String2 within String1 // by example First mark point: allow locate / find / calculate position of	2																																																											
(ii)	Value: 0 / zero Meaning: String2 not found in String1	2																																																											
(iii)	Option 1 - It is possible to “fall off” the end of String1 (or by example) while string match (for example, String1 = "Retrace", String2 = "Raced") - MID (String1, x, 1) // description of 'subscript out of range' Option 2 - If either string is empty then - subscript out of range error // description Option 3 - If String1 found within String 2 - An endless loop will occur	2																																																											



Appendix - Program Code Solutions

3 (b)(ii): VB.NET

```

Console.Write("Enter start position: ")
StartPos = Console.ReadLine()
Console.Write("Enter number to change: ")
NumToChange = Console.ReadLine()
For n = 0 To NumToChange - 1
    Console.Write("Enter new value for position: ")
    NewChar = Console.ReadLine()
    Lookup(StartPos + n) = NewChar
Next
Console.WriteLine(NumToChange & " entries changed")

```

ALTERNATIVE:

```

Console.Write("Enter start position: ")
StartPos = Console.ReadLine()
Console.Write("Enter number to change: ")
NumToChange = Console.ReadLine()
n = 0
Do
    Console.Write("Enter new value for position: ")
    NewChar = Console.ReadLine()
    Lookup(StartPos + n) = NewChar
    n = n + 1
Loop Until n = NumToChange
Console.WriteLine(NumToChange & " entries changed")

```

3 (b)(ii): Pascal

```

write('Enter start position: ');
readln(StartPos);
write('Enter number to change: ');
readln(NumToChange);
for n := 0 to NumToChange - 1 do
begin
    write('Input new value for position: ');
    readln(NewChar);
    LookUp[Startpos + n] := NewChar;
end;
writeln(IntToStr(NumToChange) + ' entries changed');

```



ALTERNATIVE:

```

write('Enter start position: ');
readln(StartPos);
write('Enter number to change: ');
readln(NumToChange);
n := 0;
repeat
    write('Input new value for position: ');
    readln(NewChar);
    LookUp[Startpos + n] := NewChar;
    n := n + 1;
until (n = NumToChange);
writeln(IntToStr(NumToChange) + ' entries changed');
```

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

```




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 6.



6 (a) (i)

n	x	Flag	m	NewString
–		TRUE	7	“”
1	‘B’	FALSE		“B”
2	‘i’			“Bi”
3	‘g’			“Big”
4	‘▽’	TRUE		“Big▽”
5	‘B’	FALSE		“Big▽B”
6	‘e’			“Big▽Be”
7	‘n’			“Big▽Ben”

One mark per correct column (**columns 3 and 4 count as 1**)

- Arrows indicate required sequence (i.e. “B” can’t be in row before TRUE)
- Letter case must be as shown
- Ignore quotation symbol
- Allow “ ” or ▽ for space symbol

(ii) (To return a string where:)

[2]

- The first character of each word is capitalised / made upper case // by explanation
- The remaining characters (of each word) are made lower case

(b) (i) • The function operates normally // returns an ‘empty string’

[1]

(ii) Examples of suitable test strings:

[3]

- Strings with all capitals
- Strings with all lower case
- Strings with first letters of words already capitalised (i.e. in correct format)
- Strings in “reverse” format – i.e. first letters lower case, the rest upper case
- String with only one word
- Strings with multiple spaces
- Strings with numbers / symbols

One mark for each string example **plus** supporting explanation.



Appendix – Program Code Example Solutions

Q3 (b): VB.NET

```

Console.WriteLine("Enter start position")
StartIndex = Console.ReadLine()
Console.WriteLine("Enter how many")
NumberToOutput = Console.ReadLine()
For Index = StartIndex To StartIndex + NumberToOutput - 1
    OriginalChar = Chr(Index)
    CipherChar = Lookup(Index)
    Console.WriteLine("Index " & Index & ": Character " & OriginalChar &
        " has substitute character " & CipherChar)
Next Index

```

Q3 (b): Pascal

```

Writeln('Enter start position');
Readln(StartIndex);
Writeln('Enter how many');
Readln(NumberToOutput);
For index := StartIndex To StartIndex + NumberToOutput - 1 Do
Begin
    OriginalChar := chr(index);
    CipherChar := Lookup[index];
    writeln("Index " + index + ": Character " + OriginalChar +
        " has substitute character " & CipherChar);
end;

```

Q3 (b): Python

```

startIndex = int(input("enter start position"))
numberToOutput = int(input("enter how many"))
for index in range(startIndex, (startIndex + numberToOutput)) :
    OriginalChar = chr(index)
    CipherChar = Lookup[index]
    print("Index " + (index) + ": Character " + OriginalChar +
        " has subst char " + CipherChar)

```



Q5 (b): VB.NET

A StreamReader() solution:

```
Sub OutPutLocationList()

    Dim Total As Integer
    Dim FileData As String
    Dim ObjReader As IO.StreamReader
    ObjReader = New IO.StreamReader("C:\MyMusic.txt")
    Dim CDLocation As String
    Dim CDTitle As String
    Dim CDArtist As String

    Total = 0
    Console.WriteLine("Input location to search ")
    CDSearch = Console.ReadLine
    Do While ObjReader.Peek <> -1
        FileData = ObjReader.ReadLine()
        If Right(FileData, 8) = CDLocation Then
            CDTitle = Left(FileData, 40)
            CDArtist = Mid(FileData, 41, 40)
            Console.WriteLine(CDTitle & " " & CDArtist)
            Total = Total + 1
        End If
    Loop
    Console.WriteLine(Total & " CDs were found")
    ObjReader.Close()
End Sub
```

A legacy FileOpen() solution:

```
Sub OutPutLocationList()

    Dim Total As Integer
    Dim FileData As String
    FileOpen (1, "C:\MyMusic.txt", OpenMode.Input)
    Dim CDLocation As String
    Dim CDTitle As String
    Dim CDArtist As String

    Total = 0
    Console.WriteLine("Input location to search ")
    CDSearch = Console.ReadLine
    Do While NOT EOF(1)
        Input(1, FileData)
        If Right(FileData, 8) = CDSearch Then
            CDTitle = Left(FileData, 40)
            CDArtist = Mid(FileData, 41, 40)
            Console.WriteLine(CDTitle & " " & CDArtist)
            Total = Total + 1
        End If
    Loop
    Console.WriteLine(Total & " CDs were found")
    FileClose(1)
End Sub
```



Q5 (b): Pascal

```

procedure OutputLocationList;
var
    FileData, CDLocation, CDTitle, CDArtist : String;
    CDFile : Textfile;
    Total : Integer;
Begin
    Total := 0;
    Writeln('Input location to search ');
    Readln(CDSearch);
    AssignFile(CDFile, 'MyMusic.txt');
    Reset(CDFile);

    While not eof(CDFile) Do
    Begin
        readln(CDFile, FileData);
        If copy(FileData, 80, 8) = CDSearch Then
        Begin
            CDTitle := copy(FileData, 1, 40);
            CDArtist := copy(FileData, 41, 40);
            Writeln(CDTitle + ' : ' + CDArtist);
            Total := Total + 1;
        End;
    End;

    Writeln(Total + ' CDs were found');
    CloseFile(CDFile);

End

```

Q5 (b): Python

```

#total : Integer
#CDSearch, LineOfText, LineString : String

Def OutPutLocationList():
    FileHandle = open("MyMusicPy.TXT", "r")
    total = 0
    CDSearch = input("Enter location to search")
    LineOfText = FileHandle.readline()
    while len(LineOfText) > 0:
        LineString = LineOfText[80:87]    #extract last 8 characters
        (location)
        if LineString == CDSearch:
            total = total+1
            CDTitle = LineOfText[0:39]    #extract CD title
            CDArtist = LineOfText[40:79]  #extract CD artist
            print(CDTitle + ": " + CDArtist)
            LineOfText = FileHandle.readline() #read next line
        print("There are " + str(total) + " in that location")
    FileHandle.close()

```

QUESTION 7.



2 (a)

Status2	ReadingCount	ThisBit	BitCount	OUTPUT
			0	
1	1	1	1	
	2	0	1	
	3	1	2	
	4	1	3	
	5	1	4	
	6	0	4	
1				
	1	1	5	Error – Investigate
			0	
	2	1	1	
	3	0	1	
	4	0	1	
	5	1	2	
	6	1	3	
0				

1 must 'follow' 6 as shown by arrow. Can have only 1 or nothing above.

1

0 must 'follow' 6 as shown by arrow. Can have only 1 or nothing above.

0

One mark per area outlined

[7]



(b) One mark for each of:

- Assignment: 01 // 02 // 06 // 09 // 14 // 18
- Selection: 07 // 11
- Iteration: 03 // 05

- 3 (a) (i) 7 [1]
- (ii) 103 [1]
- (iii) 'K' [1]
- (iv) "come" [1]

(b) (i) PROCEDURE CalculateCustomerID
 OUTPUT "Key in surname"
 INPUT Surname
 Length ← **CHARACTERCOUNT (Surname)**
 CustomerID ← 0
 FOR i ← 1 TO Length
 //NextChar is a single character from Surname
 Nextchar ← 1 **SUBSTR(Surname, i, 1)** // **ONECHAR(Surname, i)**
 NextCodeNumber ← ASC(NextChar)
 CustomerID ← CustomerID + **NextCodeNumber**
 ENDFOR
 OUTPUT "Customer ID is ", CustomerID
 ENDPROCEDURE

One mark per phrase in **bold** [3]



- (ii) 'Pseudocode' solution included here for development and clarification of marks. Programming language example solutions appear in the Appendix.

```

PROCEDURE CalculateCustomerID
  DECLARE Surname : STRING
  DECLARE NextChar : CHAR
  DECLARE NextCodeNumber, i, CustomerID, SLength : INTEGER
  OUTPUT "Key in surname"
  INPUT Surname
  SLength ← LEN(Surname)
  CustomerID ← 0
  FOR i ← 1 TO SLength
    //NextChar is a single character from Surname
    Nextchar ← MID(Surname, i, 1)
    NextCodeNumber ← ASC(NextChar)
    CustomerID ← CustomerID + NextCodeNumber
  ENDFOR
  OUTPUT "Customer ID is ", CustomerID
ENDPROCEDURE

```

Mark as follows:

- Declaration of Surname as STRING and NextChar as CHAR and any three INTEGERS
- Prompt and Input
- Calculation of string length
- FOR Loop to process all characters in the string
- Assignment to NextChar in a loop
- Assignment to NextCodeNumber in a loop
- Totalling CustomerID in a loop
- Output following a loop

[6]

- (c) (i) Visual Basic

Function CalculateCustomerID(ByVal AnyName AS STRING) As Integer

Pascal

FUNCTION CalculateCustomerID(AnyName : STRING) : INTEGER

Python

def CalculateCustomerID(AnyName):

Mark as follows:

- Correct keyword + Function name
- Single input parameter of correct type
- Return parameter type

[3]

- (ii) Visual Basic

Return customerID // CalculateCustomerID = CustomerID

Pascal

Result := CustomerID // CalculateCustomerID := CustomerID

Python

Return CustomerID

[1]



(iii) Visual Basic

ThisID = CalculateCustomerID ("Wilkes")

Pascal

ThisID := CalculateCustomerID ('Wilkes')

Python

ThisID = CalculateCustomerID ("Wilkes")

One mark per underlined element

[3]

- (d) (i) • Built-in functions are made available by the programming language / already in the system
 • Built-in functions are ready made and tested
 • User-defined functions can be modified // built-in cannot be modified
 • User defined functions can be designed to meet the user's requirements
 • User-defined functions can only be used in that program / module

[Max 2]

- (ii) • They have an identifier name
 • They return a value
 • They have none, one or more arguments
 • Both perform a specific task
 • Both represent re-usable code
 • Both are 'called'

[Max 2]

- 4 (a) • Create / modify the source code using the text editor
 • Compiler translates the source code
 • Compiler produces the object code

[Max 3]

- (b) (i) • Errors in keywords are highlighted // before the compilation process
 • Provides line-by-line syntax checking as code is typed in
 • Provides line number of the error
 • Display of known identifier names
 • Auto-complete
 • Colour-coding
 • Auto-indent
 • type checking
 • Subroutine parameter checking

[Max 1]

- (ii) • Set break-points
 • Single step / step into/over subroutine
 • Window to watch the changing value of variables

[Max 1]



(c) (i) OPEN "PRODUCTS" FOR READ
 $i \leftarrow 1$
 WHILE NOT EOF("PRODUCTS")

```

    READFILE ("PRODUCTS", PCode[i])
    READFILE ("PRODUCTS", PDescription[i])
    READFILE ("PRODUCTS", Temp // PRetailPrice[i])
  }

  PRetailPrice[i] ← TONUM(Temp)

```

$i \leftarrow i + 1$
 ENDWHILE

CLOSE "PRODUCTS"
 OUTPUT "Product file contents written to arrays"

One mark per bold phrase (three READFILE() counts as a single mark)

[5]

(ii) Benefit:

- The number of file read operations is reduced (by 2/3rds)
- It may use less storage / space in the file if strings are NOT fixed length
- All the data related to a single product is read at once / in one file operation / grouped together

Drawback:

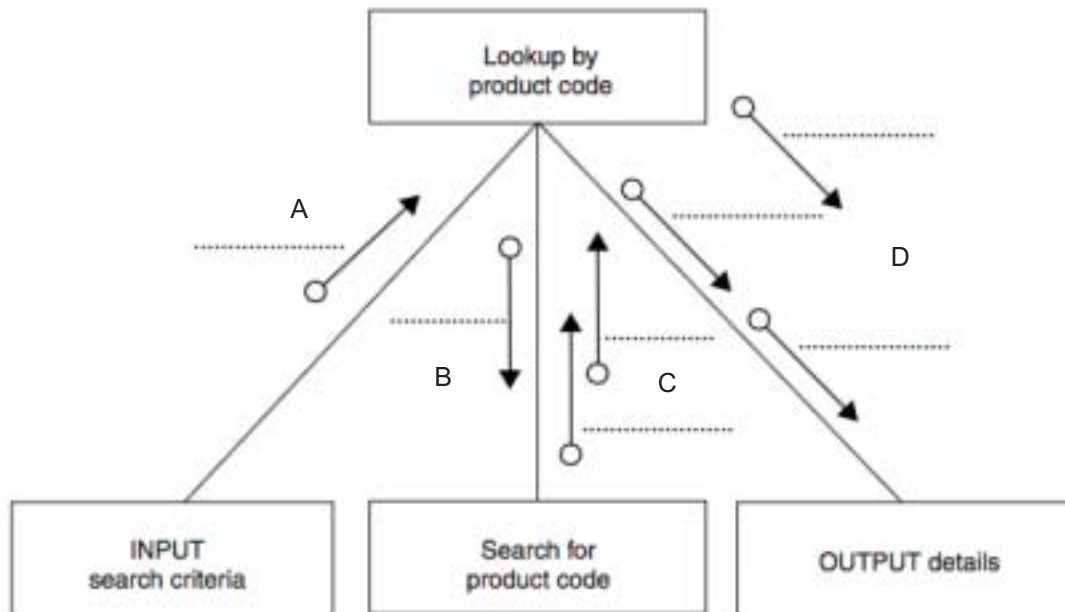
- The program will need to use the string handling functions to isolate each of the three items of data
- Difficult to isolate data items if the format is not consistent
- More difficult to search

Max one benefit and one drawback

[2]



(d)



One mark per group (one or more names) as follows:

- A: SearchCode
- B: SearchCode // ThisIndex
- C: ThisRetailPrice, ThisDescription
- D: SearchCode, ThisDescription, ThisRetailPrice

[4]



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

```

FUNCTION ProductCodeSearch(AnyName : String) RETURNS : Integer
    DECLARE FoundPos : Integer
    DECLARE i : Integer

    i ← 1
    FoundPos ← -1

    REPEAT
        IF AnyName = PCode[i]
            THEN
                FoundPos ← i
            ELSE
                i ← i + 1
            ENDIF
    UNTIL (i = 1001) OR (FoundPos <> -1)

    RETURN FoundPos

ENDFUNCTION

```

Mark as follows:

- Function header returns INTEGER
- Initialisation of index variable
- Loop through array PCode (including exit when found)
- Comparison of AnyName with PCode[i] in a loop
- Increment index variable in a loop
- Return index if AnyName found AND return -1 if AnyName not found

[Max 6]

- | | | |
|----------|---------------|-----|
| 5 | (i) 13 / 13.0 | [1] |
| | (ii) 18.6 | [1] |
| | (iii) TRUE | [1] |
| | (iv) 32 | [1] |
| | (v) 22 | [1] |

*** End of Mark Scheme – Example program code solutions follow ***



Appendix – Example program code solutions

3(b)(ii): Visual Basic

```
Dim Surname As String
Dim NextChar As Char
Dim NextCodeNumber As Integer
Dim i As Integer
Dim CustomerID As Integer
Dim SLength As Integer

Console.Write("Key in surname ")
Surname = Console.ReadLine
SLength = Len(Surname)
CustomerID = 0
For i = 1 To SLength
    \\ NextChar is a single character from surname
    NextChar = Mid(Surname, i, 1)
    NextCodeNumber = Asc(NextChar)
    CustomerID = CustomerID + NextCodeNumber
Next

Console.WriteLine("Customer ID is " & CustomerID)
```

3(b)(ii): Pascal

```
Var Surname : string;
    SLength, i, CustomerID, NextCodeNumber : integer;
    NextChar : char;
begin
    Writeln ('Enter the surname: ');
    Readln (Surname);
    SLength := Length(Surname);
    CustomerID := 0;
    For i := 1 to SLength do
        begin
            NextChar := SurName[i];
            NextCodeNumber := Ord(NextChar);
            CustomerID := CustomerID + NextCodeNumber;
        end;
    Writeln ('Customer ID is ', CustomerID);
    Readln;
end.
```

QUESTION 1.



- 6 (a) Combination of staff and task number // the pair of numbers // the pair of random numbers // there will be duplicates /repeats//some staff tasks will not be generated
- (b) (i) 04 // 03
- (ii) 27 // 28 [1]
- (iii) 20 [1]
- (iv) 11 / 12 [1]
- (c) (i) Zero [1]
- (ii) Completed <> 60 // NewStaffTask = FALSE [1]
Allow: Inclusion of the WHILE
- (iii) Determines whether this combination of StaffNum and TaskNum has been completed [1]
Assigns value TRUE if not already generated [1]
Flags that this is the first time this staff + task has been selected/to exit the loop [1]
Outputs the new staff + task number [1]
- [MAX 3]
- (iv) TaskGrid : ARRAY[1:5, 1:12] OF BOOLEAN
1 mark | 1 mark [2]
- (d) **Pseudocode ...**
(SELECT) CASE (OF) + ENDCASE using StaffNo
1 mark 1 mark
(CASE) 1: StaffName ← "Sadiq" 1
(CASE) 2: StaffName ← "Smith"
(CASE) 3: StaffName ← "Ho"
(CASE) 4: StaffName ← "Azmah"
(CASE) 5: StaffName ← "Papadopolis"
(all four cases ...) 1
ENDCASE // ENDSELECT [4]

Visual Basic

```
Select Case StaffNo
    Case 1
        StaffName = "Sadiq"
    Case 2
        StaffName = "Smith"
    Case 3
        StaffName = "Ho"
    Case 4
        StaffName = "Azmah"
    Case 5
        StaffName = "Papadopolis"
End Select
```

QUESTION 2.



- 4 (a) The combination of suit and card number // the 'pair' of numbers // the pair of numbers
There will be duplicates/repeats//not all cards will be drawn
- (b) (i) 32 // 33 [1]
(ii) 27 // 28 [1]
(iii) 08 [1]
(iv) 12 // 13 [1]
- (c) 1 [1]
- (d) DealCount <> 52 // NewCard = FALSE
Allow: Inclusion of the WHILE [1]
- (e) Test has the card has already been drawn? [1]
Set value TRUE for this card entry (in the array)/this card [1]
Flags that this is the first time this card has been drawn // decides if another card must be generated [1]
Outputs the new card value [1]
- [Max 2]
- (f) CardPack ARRAY[1:4 , 1:13] OF/:/AS BOOLEAN [1]
Allow: parentheses
- (g) **Pseudocode ...**
(SELECT) CASE (OF) CardValue + ENDCASE [1]
 (CASE) 1: CardName ← "Ace" 1 mark for any one correct [1]
 (CASE) 11: CardName ← "Jack"
 (CASE) 12: CardName ← "Queen"
 (CASE) 13: CardName ← "King" (final three cases ...) [1]
 OTHERWISE (/ELSE) CardName ← CardValue //
 (CASE) 2 TO 10: CardName ← CardValue [1]
ENDCASE // ENDSELECT

Note: Must be double quotes present and correct case

Page 5	Mark Scheme
	Cambridge International AS/A Level – October/November 2015



Visual Basic

```
Select Case CardValue
    Case 1
        CardName = "Ace"
    Case 11
        CardName = "Jack"
    Case 12
        CardName = "Queen"
    Case 13
        CardName = "King"
    Case Else // Case 2 to 10
        CardName = Str(CardValue) [4]
End Select
Allow: omission of Str
```


QUESTION 3.



Question	Answer
6(a)	Pseudocode solution included here for development and clarification of mark scheme. Programming language solutions appear in the Appendix. <pre> FUNCTION ValidatePassword(InString : STRING) RETURNS BOOLEAN DECLARE LCaseChar, UCaseChar, NumChar, n : INTEGER DECLARE NextChar : CHAR DECLARE ReturnFlag : BOOLEAN ReturnFlag ← TRUE LCaseChar ← 0, UCaseChar ← 0, NumChar ← 0 FOR n ← 1 TO LENGTH(InString) NextChar ← MID(InString,n,1) IF NextChar >= 'a' AND NextChar <= 'z' THEN LCaseChar ← LCaseChar + 1 ELSE IF NextChar >= 'A' AND NextChar <= 'Z' THEN UCaseChar ← UCaseChar + 1 ELSE IF NextChar >= '0' AND NextChar <= '9' THEN NumChar ← NumChar + 1 ELSE ReturnFlag ← False //invalid character ENDIF ENDIF ENDIF ENDIF ENDFOR IF Not (LCaseChar>=2 AND UCaseChar>= 2 AND NumChar>= 3) THEN ReturnFlag ← FALSE ENDIF RETURN (ReturnFlag) ENDFUNCTION </pre> 1 mark for each of the following: 1. Correct Function heading and ending 2. Declaring three counter variables (upper, lower, numeric) 3. Initialising counters 4. Correct loop 5. Picking up NextChar from InString 6. Check and count number of lower case 7. Check and count number of upper case



Question	Answer	
6(a)	8. Check and count number of numeric 9. Check for invalid character 10. Combine all four tests into a single Boolean value 11. Returning correct Boolean value	
6(b)(i)	String1: (e.g. "AAbb123") One mark for a valid string having: • at least 2 uppercase alphabetic • at least 2 lowercase alphabetic • at least 3 numeric characters • No other character String2 – String5: One mark for correct string and explanation (testing different rules of the function) Test strings breaking different rules: • With incorrect numbers of: • Lower case characters • Upper case characters • Numeric characters • Containing an invalid character	5
6(b)(ii)	White Box	1
6(b)(iii)	• Testing may be carried out before the modules are developed // not ready for full testing • Module stubs contain simple code to provide a known response // temporary replacement for a called module	2

**Programming Solutions****Programming Code Example Solutions****Q5 : Visual Basic**

```
Sub SearchFile()  
    Dim FileData As String  
    Dim SearchID As String  
    Dim ArrayIndex As Integer  
  
    ArrayIndex = 1  
    FileOpen(1, "LoginFile.txt", OpenMode.Input)  
    SearchID = Console.ReadLine()  
  
    Do While Not EOF(1)  
        FileData = LineInput(1)  
        If SearchID = LEFT(FileData, 5) Then  
            LoginEvents(ArrayIndex, 1) = Mid(Filedata, 6, 4)  
            LoginEvents(ArrayIndex, 2) = Right(Filedata, 14)  
            ArrayIndex = ArrayIndex + 1  
        End If  
    Loop  
    FileClose(1)  
End Sub
```

Alternative:

```
Sub SearchFile()  
    Dim FileData As String  
    Dim SearchID As String  
    Dim ArrayIndex As Integer  
    Dim MyFile As System.IO.StreamReader  
  
    ArrayIndex = 1  
    MyFile = Mycomputer.FileSystem.OpenTextFileReader("Loginfile.txt")  
    SearchID = Console.ReadLine()  
  
    Do While MyFile.Peek < > -1  
        FileData = MyFile.ReadLine()  
        If SearchID = LEFT(FileData, 5) Then  
            LoginEvents(ArrayIndex, 1) = Mid(Filedata, 6, 4)  
            LoginEvents(ArrayIndex, 2) = Right(Filedata, 14)  
            ArrayIndex = ArrayIndex + 1  
        End If  
    Loop  
    MyFile.Close  
End Sub
```

**Q5 : Pascal**

```
Procedure SearchFile();
```

```
var FileData      : String;  
var SearchID      : String;  
var ArrayRow      : Integer;  
Var MyFile        : Text;
```

```
Begin
```

```
    ArrayRow := 1;  
    Assign(MyFile, "Loginfile.txt");  
    Reset(MyFile);  
    Readln(SearchID);
```

```
    While NOT EOF(MyFile) do
```

```
    Begin
```

```
        Readln(MyFile, FileData)  
        IF SearchID = LeftStr(FileData,5) then  
            Begin  
                LoginEvents[ArrayRow,1] = Copy(FileData,6,4);  
                LoginEvents[ArrayRow,2] = Rightstr(FileData,14);  
                ArrayRow = ArrayRow + 1
```

```
            End;
```

```
    End;
```

```
        Close(MyFile);
```

```
End.
```

**Q5 : Python**

```
def SearchFile():

    # FileData : STRING
    # ArrayRow : INTEGER
    # SearchID : STRING

    ArrayRow = 0
    MyFile = open("Loginfile.txt", 'r')
    SearchID = input()
    FileData = MyFile.readline()

    While FileData != ""
        If SearchID == FileData[:5]           #First 5 characters
            LoginEvents[ArrayRow][1] = FileData[5:9]   #next 4 characters
            LoginEvents[ArrayRow][2] = FileData[-14:]  #last 14 characters
            ArrayRow = ArrayRow + 1
            FileData = MyFile.readline()

    myFile.close()
    return()
```

Alternative:

```
def SearchFile():

    # FileData : STRING
    # ArrayRow : INTEGER
    # SearchID : STRING

    ArrayRow = 0
    Myfile = open("Loginfile.txt", 'r')
    SearchID = input()
    For FileData in MyFile
        IF SearchID == FileData[:5]           #First 5 characters
            LoginEvents[ArrayRow][1] = FileData[5:9]   #next 4 characters
            LoginEvents[ArrayRow][2] = FileData[-14:]  #last 14 characters
            ArrayRow = ArrayRow + 1

    MyFile.close()
    return()
```

**Q6 (a): Visual Basic**

```
Function ValidatePassword(InString As String) As Boolean
    Dim LCaseChar, UCaseChar, NumChar As Integer
    Dim NextChar As Char
    Dim ReturnFlag As Boolean
    Dim n As Integer
    ReturnFlag = TRUE
    LCaseChar = 0
    UCaseChar = 0
    NumChar = 0

    For n = 1 to Len(InString)
        NextChar = Mid(InString, n, 1)
        If NextChar >= 'a' And NextChar <= 'z' Then
            LCaseChar = LCaseChar + 1
        Else
            If NextChar >= 'A' And NextChar <= 'Z' Then
                UCaseChar = UCaseChar + 1
            Else
                If NextChar >= '0' And NextChar <= '9' Then
                    NumChar = NumChar + 1
                Else
                    ReturnFlag = False    //invalid character
                End If
            End If
        End If
    Next

    If NOT (LCaseChar >= 2 And UCaseChar >= 2 And NumChar >= 3) Then
        ReturnFlag = FALSE
    End If

    Return(ReturnFlag)

End Function
```

**Q6 (a): Pascal**

```
Function ValidatePassword(InString : String): Boolean;
  Var LCaseChar, UCaseChar, NumChar : Integer;
  Var NextChar : Char;
  Var ReturnFlag : Boolean;
  Var n : Integer;

  begin
    ReturnFlag := TRUE;
    LCaseChar := 0;
    UCaseChar := 0;
    NumChar := 0;

    For n := 1 to Length(InString) do
      begin
        NextChar := Copy(InString,n,1);
        If NextChar >= 'a' And NextChar <= 'z' Then
          LCaseChar := LCaseChar + 1
        Else If NextChar >= 'A' AND NextChar <= 'Z' Then
          UCaseChar := UCaseChar + 1
        Else If NextChar >= '0' AND NextChar <= '9' Then
          NumChar := NumChar + 1
        Else
          ReturnFlag := False      //invalid character
      end
    end

    If NOT(LCaseChar >= 2 And UCaseChar >= 2 And NumChar >=3) then
      ReturnFlag := False;

  ValidatePassword := ReturnFlag
end;
```

**Q6 (a): Python**

```
def ValidatePassword(InString):
    # lCaseChar, uCaseChar, numChar : INTEGER
    # nextChar : CHAR
    # returnFlag : BOOLEAN
    # n : INTEGER
    returnFlag = TRUE
    lCaseChar = 0
    uCaseChar = 0
    numChar = 0

    for n in range (0, Len(InString))
        nextChar = InString[n]
        If nextChar >= 'a' and nextChar <= 'z':
            lCaseChar = lCaseChar + 1
        ELSE:
            IF nextChar >= 'A' and nextChar <= 'Z':
                uCaseChar = uCaseChar + 1
            ELSE:
                IF nextChar >= '0' and nextChar <= '9':
                    numChar = numChar + 1
                ELSE:
                    returnFlag = False    //invalid character

    IF Not (lCaseChar >= 2 and uCaseChar >= 2 and numChar >= 3):
        returnFlag = FALSE

    Return (returnFlag)

#next code block
```


QUESTION 4.



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 a process to count the number of integers between StartNumber and EndNumber (inclusive) that are divisible by Divisor. It begins with a START terminal, followed by an input block for StartNumber, EndNumber, and Divisor. Two initialization steps are shown: NumberFound ← 0 and Number ← StartNumber, which are grouped by a bracket. The process then enters a loop starting with a connector. The first step in the loop is to calculate the remainder of Number divided by Divisor using the MODULUS function. A decision diamond checks if the remainder is 0. If YES, the current Number is output, and NumberFound is incremented. If NO, the process skips the output and increment steps. Both paths lead to a second decision diamond that checks if Number equals EndNumber. If YES, the final count is output as "Count: " followed by NumberFound, and the process stops at the STOP terminal. If NO, the Number is incremented, and the loop restarts at the connector before the MODULUS calculation.



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 5.

Cambridge
International
AS & A Level

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><thead><tr><th>Data value</th><th>Data type</th></tr></thead><tbody><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></tbody></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><thead><tr><th>Data</th></tr></thead><tbody><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></tbody></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 border="1"> <thead> <tr> <th>Line number</th><th>Error</th><th>Correction</th></tr> </thead> <tbody> <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> </tbody> </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 6.



Question	Answer				
1(a)(i)	Statement		Assignment	Selection	Iteration
	CASE OF TempSensor1			✓	
	ELSE			✓	
	REPEAT				✓
	ENDFOR				✓
	DayNumber ← DayNumber + 1		✓		
	Error ← TRUE		✓		
	One mark per row				
1(b)(i)	Statement		Data type		
	Revision ← 500		INTEGER		
	FuelType ← 'P'		CHAR		
	MinValue ← -6.3		REAL		
	ServiceDue ← FALSE		BOOLEAN		
	ModelRef ← "W212DEC15"		STRING		
One mark per row					
1(b)(ii)	Expression		Evaluates to		
	"Month: " & MID(ModelRef, 5, 3)		"Month: DEC"		
	INT(MinValue * 2)		-12		
	ASC(Revision)		ERROR		
	Revision > 500		FALSE		
	ServiceDue = TRUE OR FuelType = 'P'		TRUE		
One mark per row					

5

5

QUESTION 7.



Question	Answer
6(a)	To retain data when the computer is shut down / turned off // after the program ends Accept equivalent answer.
6(b)	'Pseudocode' solution included here for development and clarification of mark scheme. Programming language example solutions appear in the Appendix. <pre> FUNCTION SearchFileNtoZ(AccNum : STRING) RETURNS BOOLEAN DECLARE FileData : STRING DECLARE Found : BOOLEAN CONSTANT SearchFile = "UserListNtoZ.txt" Found ← FALSE OPENFILE SearchFile FOR READ WHILE NOT EOF(SearchFile) AND NOT Found READFILE SearchFile, FileData IF AccNum & '*' = LEFT(FileData, LENGTH(AccNum)+ 1) THEN Found ← TRUE ENDIF ENDWHILE CLOSEFILE SearchFile RETURN Found ENDFUNCTION </pre> One mark for each of the following: 1. Function heading and ending, (ignore parameter) and returned BOOLEAN 2. File OPEN UserListNtoZ.txt in READ mode and CLOSE 3. Conditional loop repeating until EOF() or 'Found' 4. Read a line from the file in a loop 5. Compare the correct number of characters with AccNum in a loop 6. Set termination logic if found in a loop 7. Return Boolean value



Question	Answer
6(c)	<pre> PROCEDURE FindDuplicates() DECLARE Index : INTEGER DECLARE FileData : STRING DECLARE Continue : BOOLEAN DECLARE AccNum : STRING Index ← 1 // assuming array is [1:100] Continue ← TRUE OPENFILE "UserListAtoM.txt" FOR READ WHILE NOT EOF("UserListAtoM.txt") AND Continue = TRUE READFILE "UserListAtoM.txt", FileData IF MID(FileData, 7, 1) = '*' // six character reference THEN AccNum ← LEFT(FileData, 6) ELSE AccNum ← LEFT(FileData, 9) ENDIF IF SearchFileNtoZ(AccNum) = TRUE THEN IF Index = 101 // is the array already full? THEN OUTPUT "Error - Array Full" Continue ← FALSE ELSE Duplicates[Index] ← AccNum Index ← Index + 1 ENDIF ENDIF ENDIF ENDWHILE CLOSEFILE "UserListAtoM.txt" ENDPROCEDURE </pre> One mark for each of the following (max 8): 1. Declaration and Initialisation of <code>Index</code> and used to index array <code>Duplicates</code> 2. OPEN file <code>UserListAtoM.txt</code> in READ mode and CLOSE 3. Pre-Condition loop to go through the file until <code>EOF()</code> and early termination if array full 4. Read line from file and extract account number (<code>AccNum</code>) in a loop 5. Call <code>SearchFileNtoZ</code> (with <code>AccNum</code>) following an attempt at MP4 in a loop 6. Check if return value is TRUE and if so: in a loop 7. store <code>AccNum</code> in correct array element 8. increment array <code>index</code> following an attempt at MP7 9. If array overflow OUTPUT error message



Question	Answer	
6(d)(i)	<pre>PROCEDURE ClearArray(BYREF ThisArray : ARRAY, NumElements : INTEGER, InitVal : STRING) DECLARE Index : INTEGER FOR Index ← 1 TO NumElements ThisArray[Index] ← InitVal ENDFOR ENDPROCEDURE</pre> Mark as follows: • Procedure header • Loop • Assignment within loop	
6(d)(ii)	'Pseudocode' solution included here for development and clarification of mark scheme. Programming language example solutions appear in the Appendix. <pre><u>CALL ClearArray(Duplicates, 100, "Empty")</u></pre> Mark as follows: • Procedure call • Parameter list (in brackets)	2

**Program Code Example Solutions****Question 6(b): Visual Basic**

```

Function SearchFileNtoZ(ByVal SearchString As String) As Boolean
    Dim FileData As String
    Dim Found As Boolean

    Found = FALSE

    FileOpen(1, "UserListNtoZ.txt", OpenMode.Input)

    While Not EOF(1) And Not Found

        Filedata = LineInput(1)
        If SearchString & '*' = Left(FileData, Len(SearchString)+1) Then
            Found = TRUE
        End If

    End While

    FileClose(1)

    Return Found

End Function

```

Question 6(b): Pascal

```

function SearchFileNtoZ (SearchString : string): boolean;
var
    FileData : string;
    Found : boolean;
    MyFile : text;

begin
    Found := FALSE;

    assign(MyFile, "UserListNtoZ.txt");
    reset (Myfile);

    while Not EOF(MyFile) And Not Found do
        begin
            readLn(MyFile, FileData);
            if SearchString + '*' = LeftStr(FileData, length(SearchString)+1)
then
                Found := TRUE;

            end;

        close(MyFile);

        result := Found; // SearchFileB := Found;

    end;

```

**Question 6(b): Python**

```
def SearchFileNtoZ(SearchString):  
    ## FileData : String  
    ## Found : Boolean  
    ## MyFile : Text  
  
    Found = False  
  
    MyFile = open("UserListNtoZ.txt", 'r')  
    FileData = MyFile.readline()  
    while FileData != "" and not Found :  
        if SearchString + '*' == FileData[0: len(SearchString)+1]:  
            Found = True  
  
        FileData = MyFile.readline()  
  
    MyFile.close  
  
    return(Found)
```

Question 6(d)(ii): Visual Basic

Call ClearArray(Duplicates, 100, "Empty") 'Call optional

Question 6(d)(ii): Pascal

ClearArray(Duplicates, 100, 'Empty');

Question 6(d)(ii): Python

ClearArray(Duplicates, 100, "Empty")

QUESTION 8.



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><tr><th>Index</th><th>NextChar</th><th>NumDots</th><th>NumAts</th><th>NumOthers</th></tr><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></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'