

QUESTION 4.



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")
```



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

```

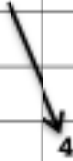
def InputData() :
    #CDTitle String (or CDTitle = "")
    #CDArtist String (or CDArtist = "")
    #CDLocation String (or CDLocation = "")

    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 5.



6 (a)	<table><tr><td>n</td><td>f</td><td>x</td><td>y</td><td>MID(String1, x, 1)</td><td>MID(String2, y, 1)</td></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> 	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'							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'																																																							
(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")
```



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.



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 7.



- 5 (a) • 2D array
• of type integer
• with identifier `PlayerScore`

[Max. 2]

- (b) (i) Stepwise refinement // Top-Down Design

[1]

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

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

```
OPENFILE "NAMES.TXT" FOR READ
i ← 1
```

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

```
CLOSEFILE "NAMES.TXT"
```

One mark for each of:

- File open in read mode
- Loop until `EOF()` or count-controlled (8 iterations)
- Read a line from the file in a loop
- Assignment to `PlayerName[1 to 8]` from the file in a loop

Close file

[Max. 4]



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

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

REPEAT
  IF ThisPlayerName = PlayerName[i]
    THEN
      Found ← TRUE
      PlayerNumber ← i
    ELSE
      i ← i + 1
  ENDIF

UNTIL (Found = TRUE) OR (i = 9)
```

One mark for each of:

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

[Max. 4]

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

[2]

(ii)

...	True
...	False

Both answers must be correct

[1]

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

Line 5:

The boundary value must be included //

`IF PlayerScore[GameIndex, PlayerIndex] >= 100 // > 99`

Line 9:

The boundary value must be included //

`IF PlayerScore[GameIndex, PlayerIndex] >= 50 // > 49`

Line 11:

One should be added to `Total50` (not `GameIndex`) //

`Total50 ← Total50 + 1`

One mark for line number + explanation

[1]

QUESTION 8.

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.



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)	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	



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 border="1"> <thead> <tr> <th data-bbox="264 1149 986 1211">Statement</th><th data-bbox="986 1149 1166 1211">White-box</th><th data-bbox="1166 1149 1347 1211">Black-box</th></tr> </thead> <tbody> <tr> <td data-bbox="264 1211 986 1312">The student does not need to know the structure of the code.</td><td data-bbox="986 1211 1166 1312"></td><td data-bbox="1166 1211 1347 1312">✓</td></tr> <tr> <td data-bbox="264 1312 986 1413">The student chooses data to test every possible path through the code.</td><td data-bbox="986 1312 1166 1413">✓</td><td data-bbox="1166 1312 1347 1413"></td></tr> <tr> <td data-bbox="264 1413 986 1514">The student chooses normal, boundary and erroneous data.</td><td data-bbox="986 1413 1166 1514">✓</td><td data-bbox="1166 1413 1347 1514">(✓)</td></tr> <tr> <td data-bbox="264 1514 986 1608">The student chooses data to test that the program meets the specification.</td><td data-bbox="986 1514 1166 1608"></td><td data-bbox="1166 1514 1347 1608">✓</td></tr> </tbody> </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.		✓	4
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.		✓															



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 9.



Question	Answer
2(a)(i)	<pre>FUNCTION CalcPoints(CardNum: STRING, Total: REAL) RETURNS INTEGER DECLARE OldPoints : INTEGER DECLARE NewPoints : INTEGER IF Total > 100 THEN OldPoints ← GetPoints(CardNum) IF OldPoints > 2000 THEN NewPoints ← INT(Total * 1.2) ELSE NewPoints ← INT(Total * 1.1) ENDIF ELSE NewPoints ← 0 ENDIF RETURN NewPoints ENDFUNCTION</pre> 1 mark for each of the following: 1 Correct FUNCTION heading (as given) and end 2 Declaring local variables for OldPoints and NewPoints 3 IF...THEN...ELSE...ENDIF with Total > 100 4 ...Newpoints set to zero if Total <= 100 5 Nested IF...THEN...ELSE...ENDIF with OldPoints > 2000 6 ... with correct assignments of NewPoints 7 Return NewPoints for all cases



Question	Answer	
2(a)(ii)	<pre> FUNCTION GetTotal() RETURNS REAL DECLARE Valid : BOOLEAN DECLARE Amount : REAL Valid ← FALSE REPEAT OUTPUT "Enter the amount" INPUT Amount IF Amount > 0 AND Amount < 10000 THEN Valid ← TRUE ENDIF UNTIL Valid = TRUE RETURN Amount ENDFUNCTION </pre> Note that the pseudocode shown is only an example. The use of an explicit flag and IF structure are not essential provided the functionality is provided. 1 mark for each of: 1 declaration of local variable(s) used 2 prompt followed by input 3 conditional loop 4 checking that input value > 0 and input value < 10000 5 returning the value	
2(b)(i)	1 mark for name, 1 mark for description Accept by example for description Name: Run-time Description: The program executes an illegal instruction // performs an illegal operation that is trapped by the OS	2
2(b)(ii)	One specific value from within each of the following ranges: For example: • 73 (a value ≤ 100) • 145.3 (a value > 100) 1 mark for the value, plus one for a meaningful description.	4

QUESTION 10.



Question	Answer
2(a)(i)	<pre>FUNCTION CalcBonus(CardNum: STRING) RETURNS INTEGER DECLARE Points : INTEGER DECLARE Bonus : INTEGER DECLARE Spend : REAL Points ← GetPoints(CardNum) Spend ← GetSpend(CardNum) IF Points > 2000 THEN Bonus ← 100 ELSE IF Spend > 1000 THEN Bonus ← 50 ELSE Bonus ← 10 ENDIF ENDIF RETURN Bonus ENDFUNCTION</pre> 1 mark for each of the following up to max 5 marks: 1 Function heading (inc parameters) and ending 2 Declaring local variables and two function calls as above 3 IF...THEN...ELSE...ENDIF with Points > 2000 4 (Nested) IF...THEN...ELSE with Spend > 1000 5 ... assignment of Bonus to 10, 50 100 6 Return parameter



Question	Answer	
2(a)(ii)	The pseudocode shown here is only an example. The use of an explicit flag and IF structure are not essential provided the functionality is provided. <pre> FUNCTION GetCardNumber() RETURNS STRING DECLARE Valid : BOOLEAN DECLARE CardNum : STRING Valid ← FALSE REPEAT OUTPUT "Enter card number" INPUT CardNum IF LENGTH(CardNum) = 16 AND IS_NUM(CardNum) = TRUE THEN Valid ← TRUE ENDIF UNTIL Valid RETURN CardNum ENDFUNCTION </pre> 1 mark for each of the following: 1 Declaring local variable to store user input 2 Conditional Loop 3 Prompt and input of CardNum 4 Length check 5 Checking IS_NUM(CardNum) is TRUE 6 Return a value	
2(b)(i)	Name: Logic (error) Description: Where the program does not behave as expected / Does not give expected result / An error in the logic of the algorithm OR Name: Run-time // execution (error) Description: The program performs an illegal operation One mark for name + one mark for corresponding description	2
2(b)(ii)	Values: any Spend value and Points > 2000 Justification: Bonus should be 100 Values: Spend > 1000 and Points <= 2000 Justification: Bonus should be 50 Values: Spend <= 1000 and Points <= 2000 Justification: Bonus should be 10 2 marks for values 2 marks for relevant and appropriate reasons	4



Question	Answer
2(c)	Name: Corrective Reason: Amend the algorithm to 'eliminate errors' Name: Adaptive Reason: In response to specification change arising from changes to business rules or environment (regulatory) Name: Perfective Reason: To make improvements to the program One mark for each name plus one mark for corresponding reason up to max 4 marks

QUESTION 11.



Question	Answer	
4(a)	<pre> PROCEDURE Button(ButtonNum : INTEGER) DECLARE Limit : INTEGER IF ButtonNum = 10 // increase volume THEN IF MaxVol = 0 THEN Limit ← 49 ELSE Limit ← MaxVol ENDIF IF VolLevel < Limit THEN VolLevel ← VolLevel + 1 ENDIF ELSE // otherwise must be ButtonNum 20 - decrease IF VolLevel > 0 THEN VolLevel ← VolLevel - 1 ENDIF ENDIF ENDIF ENDPROCEDURE </pre> Mark as follows: 1. Check if parameter value = 10 2. Check if parameter value = 20 3. Check if MaxVol = 0 4. Decrement VolLevel and ensure still in range If attempting to increase volume (parameter value was 10): 5. Increment VolLevel 6. Ensure VolLevel still in range: for both cases. i.e: VolLevel ≤ 49 (for MaxVol = 0) VolLevel ≤ MaxVol (for MaxVol > 0)	
4(b)	2 independent marks for each test: Test 1 MaxVol: 0/49 VolLevel expected: 49 Test 2 VolLevel before: 34 VolLevel expected: 34 Test 3 Parameter: 20 VolLevel expected: 0	6



Question	Answer	
4(c)(i)	One mark for type, one for description: • Type: Logical Error • Description: Program does not perform as expected • Type: Run-time error • Description: Program executes an invalid instruction / out of bounds error / attempts to divide by zero // program crashes	
4(c)(ii)	One mark per bullet point: • Tests carried out before all the modules / subroutines have been written • Simple / dummy module written to simulate / model / replace the actual module / subroutine / object • Contains an output statement // returns a fixed value to indicate that the call has been made	3