



Cambridge International AS & A Level

COMPUTER SCIENCE

9608/22

Paper 2 Written Paper

October/November 2020

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 2020 series for most Cambridge IGCSE™, Cambridge International A and AS Level and Cambridge Pre-U components, and some Cambridge O Level components.

This document consists of **24** printed pages.

PUBLISHED**Generic Marking Principles**

These general marking principles must be applied by all examiners when marking candidate answers. 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.

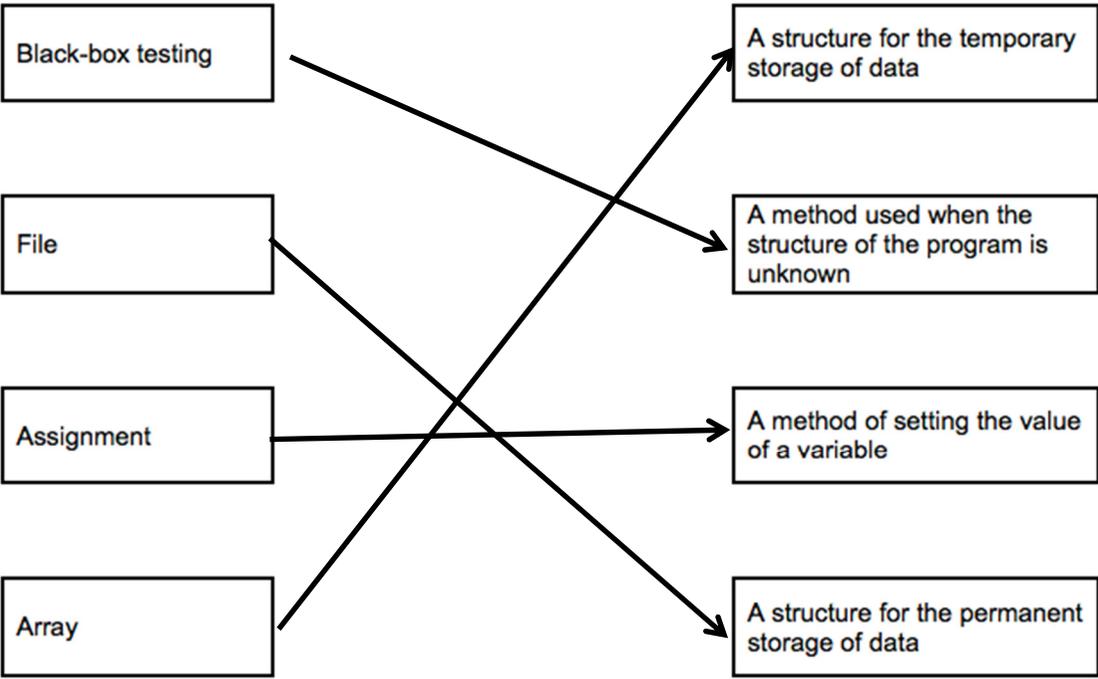
GENERIC MARKING PRINCIPLE 5:

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	Marks														
1(a)	One mark for both answers: <ul style="list-style-type: none"> • Process • Output Order not important.	1														
1(b)	One mark per bullet point (or equivalent) <p>They all represent:</p> <ul style="list-style-type: none"> • A solution to a problem / a way to perform a task • Expressed as a sequence / series of steps / stages / instructions 	2														
1(c)	1 mark per row to max 4 marks <p>Example answers:</p> <table border="1" data-bbox="353 869 996 1327"> <thead> <tr> <th data-bbox="353 869 638 933">Data type</th> <th data-bbox="638 869 996 933">Example data value</th> </tr> </thead> <tbody> <tr> <td data-bbox="353 933 638 997">BOOLEAN</td> <td data-bbox="638 933 996 997">FALSE</td> </tr> <tr> <td data-bbox="353 997 638 1061">STRING</td> <td data-bbox="638 997 996 1061">"Happy"</td> </tr> <tr> <td data-bbox="353 1061 638 1125">INTEGER</td> <td data-bbox="638 1061 996 1125">18</td> </tr> <tr> <td data-bbox="353 1125 638 1189">REAL</td> <td data-bbox="638 1125 996 1189">31234.56</td> </tr> <tr> <td data-bbox="353 1189 638 1252">CHAR</td> <td data-bbox="638 1189 996 1252">'H'</td> </tr> <tr> <td data-bbox="353 1252 638 1327">DATE</td> <td data-bbox="638 1252 996 1327">10/01/2019</td> </tr> </tbody> </table> <p>Each row must be a different data type together with an appropriate value</p>	Data type	Example data value	BOOLEAN	FALSE	STRING	"Happy"	INTEGER	18	REAL	31234.56	CHAR	'H'	DATE	10/01/2019	4
Data type	Example data value															
BOOLEAN	FALSE															
STRING	"Happy"															
INTEGER	18															
REAL	31234.56															
CHAR	'H'															
DATE	10/01/2019															

Question	Answer	Marks										
1(d)	<p>Max 3 marks, one mark for each correct line</p> <table border="0"><thead><tr><th data-bbox="371 264 645 304">Term</th><th data-bbox="1099 264 1469 304">Description</th></tr></thead><tbody><tr><td data-bbox="371 304 645 408">Black-box testing</td><td data-bbox="1099 304 1469 408">A structure for the temporary storage of data</td></tr><tr><td data-bbox="371 496 645 600">File</td><td data-bbox="1099 496 1469 600">A method used when the structure of the program is unknown</td></tr><tr><td data-bbox="371 687 645 791">Assignment</td><td data-bbox="1099 687 1469 791">A method of setting the value of a variable</td></tr><tr><td data-bbox="371 879 645 983">Array</td><td data-bbox="1099 879 1469 983">A structure for the permanent storage of data</td></tr></tbody></table> 	Term	Description	Black-box testing	A structure for the temporary storage of data	File	A method used when the structure of the program is unknown	Assignment	A method of setting the value of a variable	Array	A structure for the permanent storage of data	3
Term	Description											
Black-box testing	A structure for the temporary storage of data											
File	A method used when the structure of the program is unknown											
Assignment	A method of setting the value of a variable											
Array	A structure for the permanent storage of data											

Question	Answer	Marks										
1(e)	<p>1 mark for two rows correct, 2 marks for all rows correct.</p> <table border="1" data-bbox="353 284 1317 614"> <thead> <tr> <th data-bbox="353 284 1014 349">Expression</th> <th data-bbox="1014 284 1317 349">Evaluates to</th> </tr> </thead> <tbody> <tr> <td data-bbox="353 349 1014 414">NOT FlagB AND FlagC</td> <td data-bbox="1014 349 1317 414">TRUE</td> </tr> <tr> <td data-bbox="353 414 1014 480">NOT (FlagB OR FlagC)</td> <td data-bbox="1014 414 1317 480">FALSE</td> </tr> <tr> <td data-bbox="353 480 1014 545">(FlagA AND FlagB) OR FlagC</td> <td data-bbox="1014 480 1317 545">TRUE</td> </tr> <tr> <td data-bbox="353 545 1014 614">NOT (FlagA AND FlagB) OR NOT FlagC</td> <td data-bbox="1014 545 1317 614">TRUE</td> </tr> </tbody> </table>	Expression	Evaluates to	NOT FlagB AND FlagC	TRUE	NOT (FlagB OR FlagC)	FALSE	(FlagA AND FlagB) OR FlagC	TRUE	NOT (FlagA AND FlagB) OR NOT FlagC	TRUE	2
Expression	Evaluates to											
NOT FlagB AND FlagC	TRUE											
NOT (FlagB OR FlagC)	FALSE											
(FlagA AND FlagB) OR FlagC	TRUE											
NOT (FlagA AND FlagB) OR NOT FlagC	TRUE											

Question	Answer	Marks
2(a)	<pre> DECLARE A, B, C : INTEGER DECLARE Average : REAL INPUT A REPEAT INPUT B UNTIL B <> A REPEAT INPUT C UNTIL C <> A AND C <> B Average ← (A + B + C) / 3 OUTPUT Average IF A > B AND A > C THEN OUTPUT A ELSE IF B > A AND B > C THEN OUTPUT B ELSE OUTPUT C ENDIF ENDIF ENDIF </pre> <p>Mark as follows:</p> <ol style="list-style-type: none"> 1 Declaration of all variables used (at least A, B and C) 2 Uniqueness test on A, B and C 3 Loop(s) to repeat until three unique values have been entered 4 Calculation of average value 5 Determine the largest value 6 Output of average value and largest value 	6

Question	Answer	Marks												
2(b)	<p>One mark per correct row (Completed parts shown in bold)</p> <table border="1" data-bbox="353 300 1429 695"> <thead> <tr> <th data-bbox="353 300 1167 363">Expression</th> <th data-bbox="1167 300 1429 363">Evaluates to</th> </tr> </thead> <tbody> <tr> <td data-bbox="353 363 1167 432">"ALARM: " & RIGHT("Time: 1202", 4)</td> <td data-bbox="1167 363 1429 432">"ALARM: 1202"</td> </tr> <tr> <td data-bbox="353 432 1167 501">MID("Stepwise.", 5, 4)</td> <td data-bbox="1167 432 1429 501">"wise"</td> </tr> <tr> <td data-bbox="353 501 1167 569">1.5 * LENGTH("OnePointFive")</td> <td data-bbox="1167 501 1429 569">18</td> </tr> <tr> <td data-bbox="353 569 1167 638">NUM_TO_STRING(27.5)</td> <td data-bbox="1167 569 1429 638">"27.5"</td> </tr> <tr> <td data-bbox="353 638 1167 695">DIV(9, 4)</td> <td data-bbox="1167 638 1429 695">2</td> </tr> </tbody> </table>	Expression	Evaluates to	"ALARM: " & RIGHT ("Time: 1202", 4)	"ALARM: 1202"	MID ("Stepwise.", 5, 4)	"wise"	1.5 * LENGTH ("OnePointFive")	18	NUM_TO_STRING (27.5)	"27.5"	DIV (9, 4)	2	5
Expression	Evaluates to													
"ALARM: " & RIGHT ("Time: 1202", 4)	"ALARM: 1202"													
MID ("Stepwise.", 5, 4)	"wise"													
1.5 * LENGTH ("OnePointFive")	18													
NUM_TO_STRING (27.5)	"27.5"													
DIV (9, 4)	2													
2(c)	<p>One mark per point, example points:</p> <ol style="list-style-type: none"> 1 Subtasks make the solution more manageable // make the algorithm easier to follow 2 A subtask makes the problem easier to solve / design / program than the whole task 3 A subtask is useful when a part of the algorithm is repeated 	3												

Question	Answer	Marks
3(a)	<p>'Pseudocode' solution included here for development and clarification of mark scheme. Programming language example solutions appear in the Appendix.</p> <pre> FUNCTION CheckSkid() RETURNS BOOLEAN DECLARE Rot : ARRAY[1:4] OF INTEGER DECLARE Average : REAL DECLARE ThisRot : INTEGER DECLARE Danger : BOOLEAN FOR Index ← 1 TO 4 REPEAT OUTPUT "Input Rotation speed for wheel ", Index INPUT ThisRot UNTIL ThisRot >= 0 AND ThisRot <= 1000 Rot[Index] ← ThisRot ENDFOR Average ← (Rot[1] + Rot[2] + Rot[3] + Rot[4]) / 4 Danger ← FALSE FOR Index ← 1 TO 4 IF Rot[Index] > (Average * 1.1) OR Rot[Index] < (Average * 0.9) THEN Danger ← TRUE ENDIF ENDFOR IF Danger = TRUE THEN OUTPUT "Skid Danger" ENDIF RETURN Danger ENDFUNCTION </pre>	8

Question	Answer	Marks																								
3(a)	1 mark for each of the following: 1 Function heading and ending 2 Declare local integers for 4 rotation values and a real for the average / tolerance 3 Prompt and input four rotation values 4 Validate each input value in a loop 5 Calculate average rotation AND calculate acceptable max and min (or single tolerance, or alternative method) 6 Compare rotational value of each wheel 7 Test if rotational value of (each) wheel is within the acceptable range 8 Output a warning message and return the correct value in all cases																									
3(b)	Example answers: Test1 – No Skid detected <table border="1" data-bbox="353 708 958 839"> <thead> <tr> <th>Value 1</th> <th>Value2</th> <th>Value 3</th> <th>Value 4</th> </tr> </thead> <tbody> <tr> <td>100</td> <td>100</td> <td>100</td> <td>100</td> </tr> </tbody> </table> One of: Test2 – Skid detected (one wheel too fast) <table border="1" data-bbox="353 992 958 1123"> <thead> <tr> <th>Value 1</th> <th>Value2</th> <th>Value 3</th> <th>Value 4</th> </tr> </thead> <tbody> <tr> <td>100</td> <td>100</td> <td>100</td> <td>160</td> </tr> </tbody> </table> Test2 – Skid detected (one wheel too slow) <table border="1" data-bbox="353 1216 958 1347"> <thead> <tr> <th>Value 1</th> <th>Value2</th> <th>Value 3</th> <th>Value 4</th> </tr> </thead> <tbody> <tr> <td>100</td> <td>100</td> <td>100</td> <td>40</td> </tr> </tbody> </table> Independent marks: one mark each for Test1 and Test 2	Value 1	Value2	Value 3	Value 4	100	100	100	100	Value 1	Value2	Value 3	Value 4	100	100	100	160	Value 1	Value2	Value 3	Value 4	100	100	100	40	2
Value 1	Value2	Value 3	Value 4																							
100	100	100	100																							
Value 1	Value2	Value 3	Value 4																							
100	100	100	160																							
Value 1	Value2	Value 3	Value 4																							
100	100	100	40																							

Question	Answer	Marks
4(a)	<p>Mark as follows:</p> <ul style="list-style-type: none"> • One mark for START and END • One mark per area outlined <p>Outputs from conditional diamond must have at least one label</p>	7

Question	Answer	Marks																																																																																																																														
4(b)(i)	<p>One mark per region as indicated.</p> <table border="1" data-bbox="394 280 1433 1340"> <thead> <tr> <th>String 1</th> <th>String 2</th> <th>Len1</th> <th>RetFlag</th> <th>x</th> <th>Len 2</th> <th>NextChar</th> <th>New</th> <th>y</th> </tr> </thead> <tbody> <tr> <td>"SUB"</td> <td>"BUS"</td> <td>3</td> <td>TRUE</td> <td>1</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td>3</td> <td>'S'</td> <td>""</td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>"B"</td> <td>1</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>"BU"</td> <td>2</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>3</td> </tr> <tr> <td></td> <td>"BU"</td> <td></td> <td></td> <td>2</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td>2</td> <td>'U'</td> <td>""</td> <td>1</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>"B"</td> <td>2</td> </tr> <tr> <td></td> <td>"B"</td> <td></td> <td></td> <td>3</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td>1</td> <td>'B'</td> <td>""</td> <td>1</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td>""</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>	String 1	String 2	Len1	RetFlag	x	Len 2	NextChar	New	y	"SUB"	"BUS"	3	TRUE	1										3	'S'	""									"B"	1								"BU"	2									3		"BU"			2										2	'U'	""	1								"B"	2		"B"			3										1	'B'	""	1											""																	5
String 1	String 2	Len1	RetFlag	x	Len 2	NextChar	New	y																																																																																																																								
"SUB"	"BUS"	3	TRUE	1																																																																																																																												
					3	'S'	""																																																																																																																									
							"B"	1																																																																																																																								
							"BU"	2																																																																																																																								
								3																																																																																																																								
	"BU"			2																																																																																																																												
					2	'U'	""	1																																																																																																																								
							"B"	2																																																																																																																								
	"B"			3																																																																																																																												
					1	'B'	""	1																																																																																																																								
	""																																																																																																																															

PUBLISHED

Question	Answer	Marks
4(b)(ii)	TRUE	1
4(b)(iii)	<p>One mark for explanation of problem, one mark for test strings</p> <p>Problem:</p> <ul style="list-style-type: none"> • The inner FOR loop removes ALL characters from String2 that match the current character from String1 and not just one instance <p>Test Strings:</p> <ul style="list-style-type: none"> • 'SAME' and 'MASS' (for example) 	2
4(b)(iv)	The inner FOR loop should only remove <u>one</u> instance of the character from String2	1
4(b)(v)	<ul style="list-style-type: none"> • Dry run // White-box testing 	1
4(b)(vi)	<p>Max 2 marks, features include:</p> <ul style="list-style-type: none"> • Single stepping • Breakpoints • Variable and expressions report window • <u>Syntax</u> error highlighting 	2

Question	Answer	Marks
5(a)	<pre>PROCEDURE InitArrays() DECLARE Index : INTEGER FOR Index ← 1 TO 10000 TagString[Index] ← "" TagCount[Index] ← 0 ENDFOR ENDPROCEDURE</pre> <p>1 mark for each of the following:</p> <ol style="list-style-type: none">1 Procedure heading and ending (as shown)2 Declaration of <code>Index</code> (e.g.) as integer3 Loop for 10000 iterations4 Initialise <code>TagString</code> element to ""	4

Question	Answer	Marks
5(b)	<pre> FUNCTION SaveArrays() RETURNS INTEGER DECLARE Index, NumUnused : INTEGER DECLARE FileString : STRING CONSTANT COMMA = ',' NumUnused ← 0 OPEN "Backup.txt" FOR WRITE FOR Index ← 1 to 10000 IF TagString[Index] <> "" THEN FileString ← TagString[Index] & COMMA & NUM_TO_STRING(TagCount[Index]) WRITEFILE "Backup.txt", FileString ELSE NumUnused ← NumUnused + 1 ENDIF ENDFOR CLOSEFILE "Backup.txt" RETURN NumUnused ENDFUNCTION </pre> <p>1 mark for each of the following:</p> <ol style="list-style-type: none"> 1 Function heading and ending 2 Open the file Backup.txt in write mode and close file 3 Loop through 10000 elements 4 Test if TagString[Index] is "" in a loop 5 If not then form FileString from array elements with separator and using NUM_TO_STRING() in a loop 6 Write string to file in a loop 7 Count the number of unused elements 8 Return NumUnused not in a loop 	8

Question	Answer	Marks
5(c)	<p>'Pseudocode' solution included here for development and clarification of mark scheme. Programming language example solutions appear in the Appendix. Max 8 marks from 9 available mark points</p> <pre> FUNCTION LoadArrays() RETURNS INTEGER DECLARE ArrayIndex, Index, CountLen, Count : INTEGER DECLARE FileString, HashTag : STRING CONSTANT COMMA = ',' ArrayIndex ← 0 // first element OPEN "Backup.txt" FOR READ WHILE NOT EOF("Backup.txt") READFILE "Backup.txt", FileString Index ← 1 HashTag ← "" WHILE MID(FileString, Index, 1) <> COMMA // hashtag HashTag ← HashTag & MID(FileString, Index, 1) Index ← Index + 1 ENDWHILE TagString[ArrayIndex] ← HashTag CountLen ← LENGTH(FileString) - LENGTH(HashTag) - 1 Count ← STR_TO_NUM(RIGHT(FileString, CountLen)) // count TagCount[ArrayIndex] ← Count ArrayIndex ← ArrayIndex + 1 ENDWHILE CLOSE "Backup.txt" RETURN ArrayIndex ENDFUNCTION </pre>	8

PUBLISHED

Question	Answer	Marks
5(c)	<p>1 mark for each of the following:</p> <ol style="list-style-type: none"> 1 Function heading and ending 2 Declare and initialise <code>ArrayIndex</code> (or equivalent name) 3 Open the file <code>Backup.txt</code> in read mode and close the file 4 Loop until end of the <code>Backup.txt</code> file // string read is null 5 Read a line from the file in a loop 6 Extract hashtag and count in a loop 7 Store hashtag in <code>TagString</code> array and count in <code>TagCount</code> array after type conversion 8 Increment <code>ArrayIndex</code> in a loop 9 Return number of array elements 	

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

Appendix: Program Code Example Solutions**Q3 (a): Visual Basic**

```
Function CheckSkid() As Boolean

    Dim Rot(3) As Integer
    Dim Average As Double
    Dim ThisRot As Integer
    Dim Danger As Boolean

    For Index = 0 To 3
        Do
            Console.WriteLine("Enter Wheel Rotation Speed: ")
            ThisRot = Console.ReadLine()
            Loop Until ThisRot >= 0 And ThisRot <= 1000
            Rot(Index) = ThisRot
        Next
    Next

    Average = (Rot(0) + Rot(1) + Rot(2) + Rot(3)) / 4

    Danger = FALSE
    For Index = 0 TO 3
        If Rot(Index) > (Average * 1.1) OR Rot(Index) < (Average * 0.9) Then
            Danger = TRUE
        End If
    Next

    If Danger = TRUE Then
        Console.WriteLine("Skid Danger")
    Else
        Console.WriteLine("No Skid Danger")
    End if

    RETURN Danger

End Function
```

Q3 (a): Pascal

```
Function CheckSkid() : Boolean;

var
  Rot : array [1..4] of integer;
  Average : Real;
  ThisRot : Integer;
  Index : Integer;
  Danger : Boolean;

For Index := 1 to 4 do
  begin
    repeat
      write('Enter rotation speed : ');
      readln(ThisRot);
    until (ThisRot >= 0) And (ThisRot <= 1000);
    Rot[Index] := ThisRot;
  end;

Average := (Rot[1] + Rot[2] + Rot[3] + Rot[4]) / 4;

Danger := FALSE;
For Index := 1 to 4 do
  begin
    If (Rot[Index] > (Average * 1.1)) OR (Rot[Index] < (Average * 0.9)) then
      Danger := TRUE;
  end;

If Danger = TRUE then
  writeln('Skid Danger')
Else
  writeln('No Skid Danger');

CheckSkid := Danger;

end;
```

Q3 (a): Python

```
def CheckSkid():

    # Rot[3] As Integer
    # Average As Real
    # ThisRot As Integer
    # Danger As Boolean

    Rot = [0, 0, 0, 0]
    for Index in range(0, 4):
        while True:
            ThisRot = float(input("Enter the rotation speed of the wheel: "))
            if ThisRot >= 0 and ThisRot <= 1000:
                break
            Rot[Index] = ThisRot
    Next

    Average = (Rot[0] + Rot[1] + Rot[2] + Rot[3]) / 4

    Danger = False
    for Index in range(0, 4):
        if Rot[Index] > (Average * 1.1) or Rot[Index] < (Average * 0.9):
            Danger = True

    If Danger == True:
        print("Skid Danger")
    else:
        print("No Skid Danger")

    return Danger
```

Q5 (c): Visual Basic

```
Function LoadArrays () As Integer
```

```
    Dim ArrayIndex, Index, CountLen, Count As Integer
    Dim FileString, HashTag As String
    Dim File As New StreamReader("Backup.txt")
```

```
    Const COMMA = ','
```

```
    ArrayIndex = 0 ' First element
```

```
    Do While File.Peek <> -1
```

```
        FileString = File.ReadLine()
```

```
        Index = 1
```

```
        HashTag = ""
```

```
        Do While Mid(FileString, Index, 1) <> COMMA ' the hashtag
```

```
            HashTag = HashTag & MID(FileString, Index, 1)
```

```
            Index = Index + 1
```

```
        Loop
```

```
        TagString(arrayIndex) = HashTag
```

```
        CountLen = Len(fileString) - Len(HashTag) - 1
```

```
        Count = CInt(Right(FileString, CountLen)) ' the count
```

```
        TagCount(ArrayIndex) = Count
```

```
        ArrayIndex = ArrayIndex + 1
```

```
    Loop
```

```
    File.Close
```

```
    Return ArrayIndex
```

```
End Function
```

Q5 (c): Pascal

```
Function LoadArrays () : Integer;

var
  ArrayIndex, Index, CountLen, Count : Integer;
  FileData, HashTag : String;
  Backup : Textfile;

const
  COMMA = ',';

begin
  assignfile(Backup, 'Backup.txt');
  reset(File);

  ArrayIndex := 0; //First element

  while not EOF(File) do
    begin
      readln(Backup, FileData);
      Index := 1;
      HashTag := "";
      while midstr(FileData, Index, 1) <> COMMA do           // the hashtag
        begin
          HashTag := HashTag + midstr(FileData, Index, 1);
          Index := Index + 1;
        end;

      TagString[ArrayIndex] := HashTag;
      CountLen := length(FileData) - length(HashTag) - 1;
      Count := strtoint(RightStr(FileData, CountLen));      // the count
      TagCount[ArrayIndex] := Count;
      ArrayIndex := ArrayIndex + 1;
    end;

  closefile(File);
```

```
LoadArrays := ArrayIndex;  
end;
```

Q5 (c): Python

```
def LoadArrays ():

    # ArrayIndex, Index, CountLen, Count As Integer
    # FileString, HashTag As String
    # File As StreamReader("Backup.txt")

    COMMA = ','

    File = open("Backup.txt", "r")
    ArrayIndex = 0 #First element

    for FileString in File:
        Index = 0
        HashTag = ""
        while FileString[Index] != COMMA:           # the hashtag
            HashTag = HashTag + FileString[Index]
            Index = Index + 1

        TagString[ArrayIndex] = HashTag
        Count = int(FileString[Index+1:])          # the count
        TagCount[ArrayIndex] = Count
        ArrayIndex = ArrayIndex + 1

    File.close()

    return ArrayIndex
```