Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9608/23**

Paper 2 Fundamental Problem-Solving and Programming Skills **October/November 2021**

MARK SCHEME

Maximum Mark: 75

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:

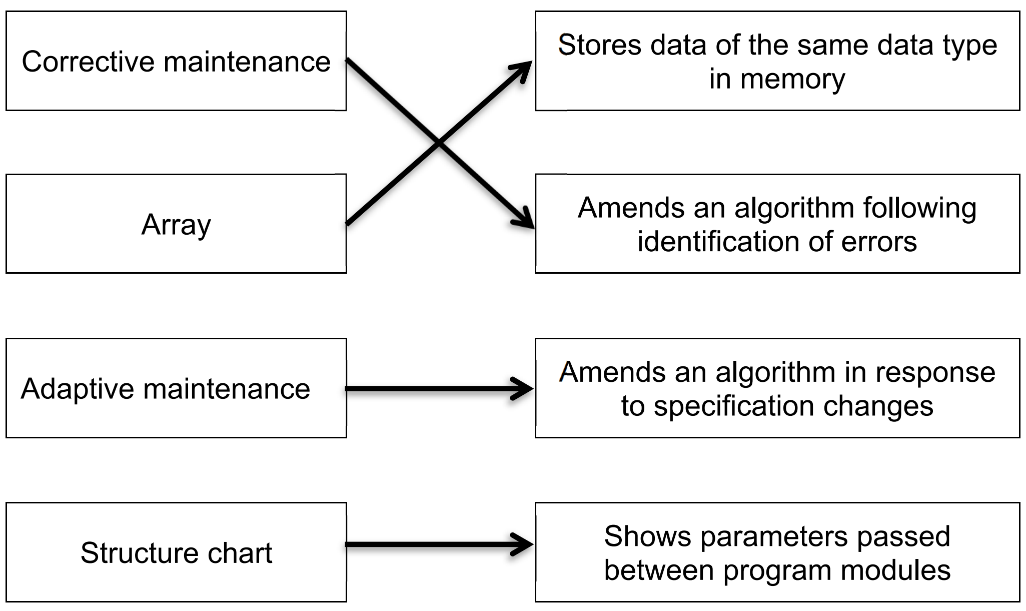
Rules must be applied consistently, e.g. in situations where candidates have not followed instructions or in the application of generic level descriptors.

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)(i)	One mark for each bullet point to Max 2 • `OUTPUT "Enter product number: "` • `INPUT ProductNumber`	**2**
1(a)(ii)	One mark for: • Store / write the product number / data // retrieve data at a later date	**1**
1(b)	One mark for each bullet point to Max 2 • (Visual method) making it easier to understand / representing / follow the logic of program structures / algorithm • Overview of the process, allowing logical errors to be identified • Provides documentation for other programmers / non-programmers	**2**
1(c)	**Technical term** **Description**  One mark for 2 correct Two marks for 3 correct	**3**
1(d)	One mark for each correct row	**5**

Corrective maintenance
Array
Adaptive maintenance
Structure chart
Stores data of the same data type in memory
Amends an algorithm following identification of errors
Amends an algorithm in response to specification changes
Shows parameters passed between program modules

Pseudocode expression	Evaluates to
`EndOfYear * Limit / Mileage`	`100`
`LENGTH(Description) / NUM_TO_STRING(Limit)`	`ERROR`
`MOD(20, LENGTH(Destination))`	`2`
`(EndOfYear < 2) AND (Limit = 20000) AND NOT(Overdue)`	`TRUE`
`MID(Description, 1, 5) & LEFT(Destination, 2) & NUM_TO_STRING(Limit / 1000)`	`"CONCREL20"`

Question	Answer	Marks

2(a)	One mark for each correct row
	
	Line number | **Corrected pseudocode statement**
	`04` / `06` | `Procedure SetOut()//` (No parameter as size is input) / `INPUT (Size)` (Add line 6)
	`05` | `DECLARE Index : ` **`INTEGER`**
	`07` | `FOR Index ← ` **`1`** ` TO 50`
	`08` | `IF Size = CarSize[Index] AND Available[Index] = ` **`TRUE`**
		4

2(b)	One mark for each bullet point • Loop from 1 to 50 / loop for all the number of cars the company owns • Compare the input / parameter value to the array `CarSize [counter]` • … **and** the corresponding element in the array `Available` is set to `TRUE` / the car is available • If so, output the current loop counter (the car number), the (parameter) `Size` followed by `"Available"`	**4**
2(c)	See below	**3**

Line number	Corrected pseudocode statement
04 06	`Procedure SetOut()//` (No parameter as size is input) `INPUT (Size)` (Add line 6)
05	`DECLARE Index : ` **`INTEGER`**
07	`FOR Index ← ` **`1`** ` TO 50`
08	`IF Size = CarSize[Index] AND Available[Index] = ` **`TRUE`**

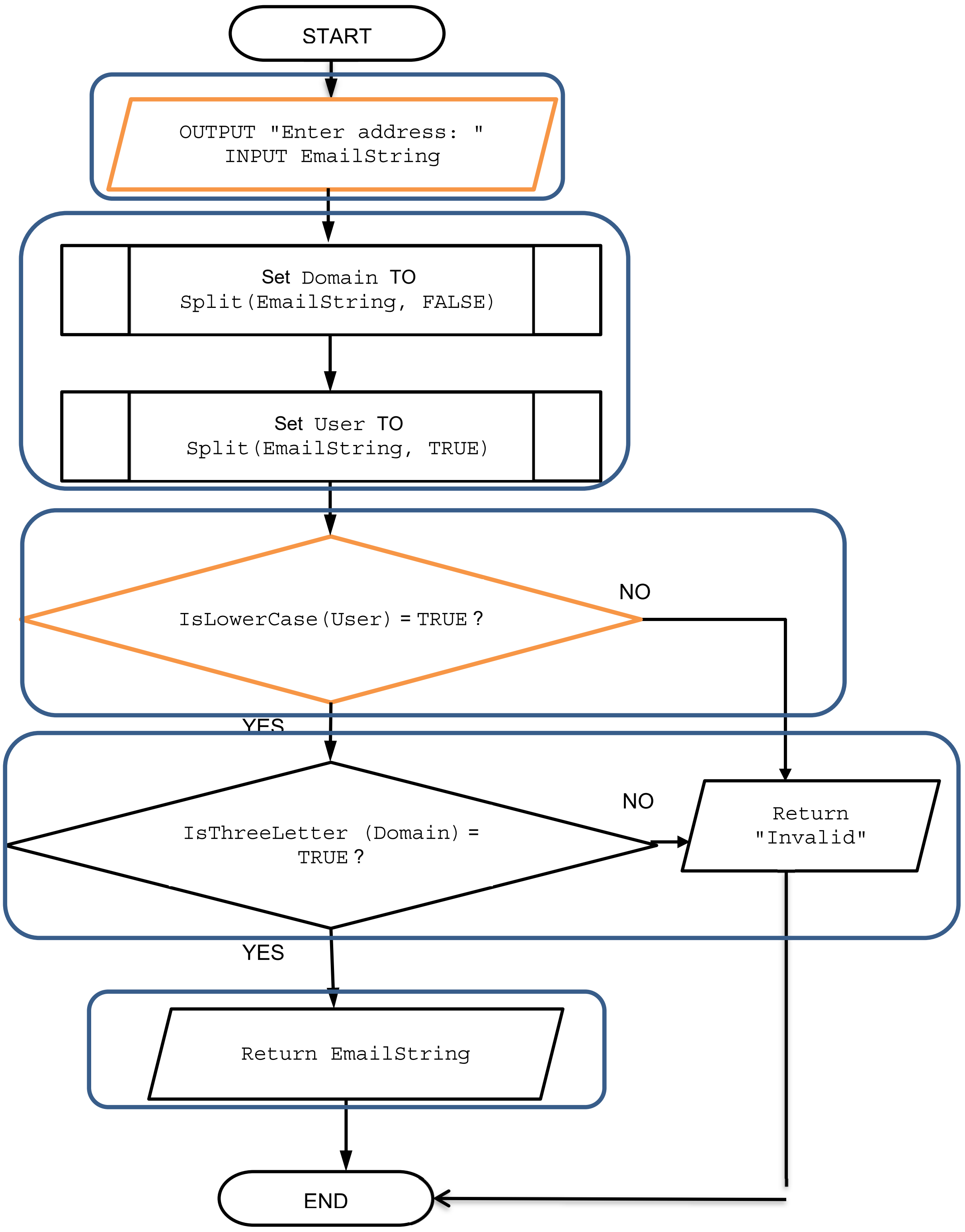
2(c)

```
FUNCTION PayUsingAccount(Balance : REAL,
         CostOfHire : REAL) RETURNS BOOLEAN

   IF CostOfHire > Balance
     THEN
       RETURN FALSE
     ELSE
       RETURN TRUE
   ENDIF

ENDFUNCTION
```

One mark for each of the following:

1 Function heading and ending including parameters and CostOfHire as REAL.
2 If CostOfHire > Balance
3 Return of True / False

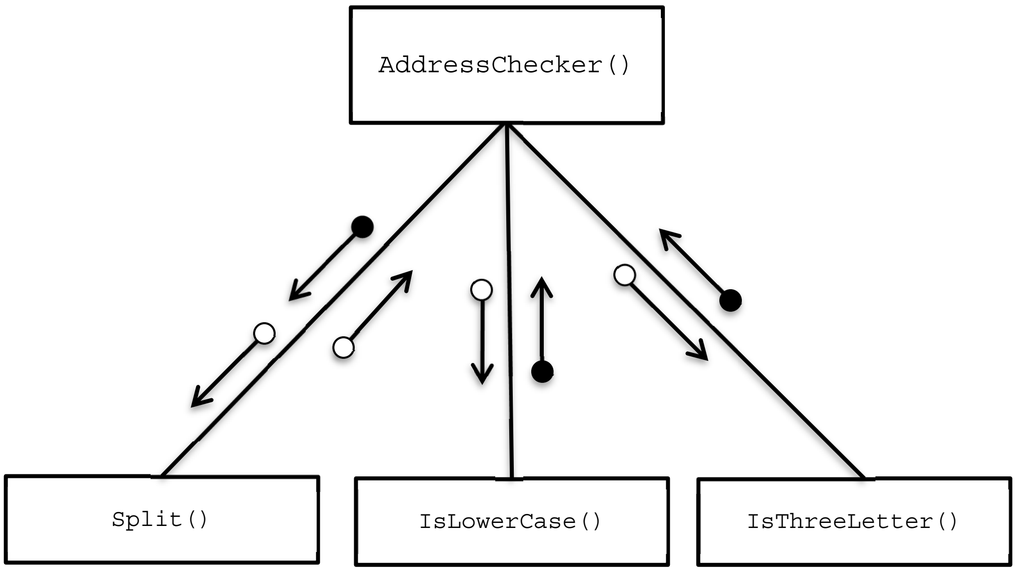
Question	Answer	Marks
3(a)	One mark for Start **and** Stop symbols One mark for each outlined group 1 (Prompt and) input `EmailString` 2 Call to `Split()` **twice** to obtain `Domain` and `User` substrings 3 Call to `IsLowerCase()` **in decision box** with `User` parameter 4 Call to `IsThreeLetter()` **in decision box** with `Domain` parameter and return `"Invalid"` if both false 5 Return `EmailString` only if all decisions evaluate to `TRUE` MPs 4 and 5 may be combined using AND	**6**

Question	Answer	Marks
		

START
OUTPUT "Enter address: "
INPUT EmailString
Set Domain TO Split(EmailString, FALSE)
Set User TO Split(EmailString, TRUE)
IsLowerCase(User) = TRUE ?
NO
YES
IsThreeLetter (Domain) = TRUE ?
NO
Return "Invalid"
YES
Return EmailString
END

Question	Answer	Marks
3(b)	One mark for each bullet point to Max 2. • `StartLine/ LastLine` is out of range • `StartLine` is after `LastLine //` • `LastLine` is before `StartLine` • `LastLine` does not exist	**2**
3(c)(i)	One mark for each bullet point Explanation • An error in the grammar of the programming language • An error that breaks the rules of the programming language	**2**

3(c)(ii)	One mark for each bullet point to Max 1
	Examples are shown in pseudocode for reference only.
	• `DECLARE NUMBER AS INTEGER`
	`NUMBER ← "LEFT"`
	• `IF …`
	`ENDWHILE`
	• `Flag ← TRUE + 3`
		1

3(d)	One mark for each bullet point • Both module names: `Split()` and `IsLowercase()` • Two parameters to `Split()` and one parameter returned from `Split()` (as shown) • All remaining parameters  AddressChecker() Split() IsLowerCase() IsThreeLetter()	**3**

Question	Answer	Marks
3(e)	One mark for the feature. One mark for the description of the feature to Max 4. • Feature: Single stepping. Description: Execute each instruction one at a time. • Feature: Breakpoints. Description: Line/lines in the program code at which point the program stops execution. • Feature: Variable / expression watch windows. Description: Windows that display the values assigned to variables/expressions as the program executes.	**4**

Question	Answer	Marks
4(a)	One mark for each column.	**5**

FileLine	Counter	LEFT(FileData, LENGTH(Match))	SubMatch = Match	Result
1	0			FALSE
2	0	"XD43688"	FALSE	(FALSE)
3	1	"TG12367"	TRUE	(FALSE)
4	1	"HD44356"	FALSE	(FALSE)
5	2	"TG12367"	TRUE	TRUE

Question	Answer	Marks
4(b)(i)	One mark for each underlined part	**2**

```
10   FUNCTION Extract(FileName:STRING, Match:STRING)
                                   RETURNS INTEGER
```

Question	Answer	Marks
4(b)(ii)	One mark for each underlined part.	**3**

```
23      WHILE NOT EOF(FileName) AND FileLine <= 100
24         READFILE FileName, FileData
25         SubMatch ← RIGHT(FileData, 10)
26         IF SubMatch = Match
27            THEN
28               Original[Counter] ← LEFT(FileData, 7)
29                 Backup[Counter] ← RIGHT(FileData, 4)
29               Counter ← Counter + 1
30         ENDIF
31         FileLine ← FileLine + 1
32      ENDWHILE
```

Question	Answer	Marks

4(c)	'Pseudocode' solution included here for development and clarification of mark scheme.
	Programming language example solutions appear in the Appendix.
	
	`PROCEDURE Encrypt()`
	`   DECLARE UnFileData : STRING // unencrypted`
	`   DECLARE EnFileData : STRING // encrypted`
	`   DECLARE Key, Counter : INTEGER`
	`   DECLARE EnChar : CHAR`
	
	`   OPENFILE "DATA.txt" FOR READ`
	`   OPENFILE "DATA-EN.txt" FOR APPEND`
	
	`   WHILE NOT EOF("DATA.txt")`
	`      READFILE "DATA.txt", UnFileData`
	`      Key ← STR_TO_NUM(MID(UnFileData, 9, 2))`
	`      EnFileData ← ""`
	
	`      FOR Counter ← 1 TO LENGTH(UnFileData)`
	`         EnChar ← CHR(Key + ASC(Mid(UnFileData, Counter,`
	`                       1)))`
	`         EnFileData ← EnFileData & EnChar`
	`      ENDFOR`
	
	`      WRITEFILE "DATA-EN.txt", EnFileData`
	
	`   ENDWHILE`
	`   CLOSEFILE "DATA.txt"`
	`   CLOSEFILE "DATA-EN.txt"`
	
	`ENDPROCEDURE`
	
	One mark for each of the following to Max 6
	
	1 Procedure heading (and ending)
	2 Open **and** close the unencrypted file for `READ` and the encrypted file for **`APPEND`**
	3 Outer conditional loop until `EOF()` of the unencrypted file.
	4 Extract the encryption key **in the outer loop**
	5 Nested inner (count-controlled loop) that generates the ASCII value of the unencrypted character
	6 **and** generates the encrypted character using CHR function **and** concatenate to form encrypted string
	7 write the encrypted string to the data file **in the outer loop**
		6

Question	Answer	Marks

5(a)	
	```
	DECLARE GeoData : ARRAY[1:20000] OF STRING
	DECLARE Review : ARRAY[1:20000] OF STRING

	FUNCTION CheckReview(UserID : STRING) RETURNS STRING

	   DECLARE Status, ReviewString, RestGeo : STRING
	   DECLARE Index, GeoCounter : INTEGER

	   Status ← "LOCATION NOT FOUND" // default to not found
	   GeoCounter ← 1
	   ReviewString  ← ""
	   RestGeo ← ""

	   Index ← 20000            // get the latest review
	   WHILE Index >= 1 AND ReviewString = ""
	     IF LEFT(Review[Index], 4) = UserID
	        THEN
	          ReviewString← Review[Index] //save review
	                                  // string for that user
	      ENDIF
	      Index ← Index - 1
	   ENDWHILE

	   IF LENGTH(ReviewString = 12 // check whether a review
	                              // exists
	     THEN
	       Status ← "NO REVIEW"
	     ELSE
	       // get the geocode of the review
	       RestGeo ← MID(ReviewString, 6, 7)
	   ENDIF

	   // find this geocode in the array GeoData
	   WHILE Status = "LOCATION NOT FOUND" AND
	                GeoCounter <= 20000
	      IF GeoData[GeoCounter] = RestGeo
	        THEN
	          Status ← RIGHT(ReviewString,
	                        LENGTH(ReviewString) - 13)
	      ENDIF
	      GeoCounter ← GeoCounter + 1
	   ENDWHILE

	   RETURN Status
	ENDFUNCTION
	```
		8

Question	Answer	Marks
5(a)	One mark for each of the following to max 8 1 Declaration of local variables (but not `UserID` which must be parameter or arrays) 2 **Conditional** loop iterating through the `Review` array finding the latest review 3 ... extract first 4 characters from `Review` array and compare with parameter `UserID` 4 … if True, check if a review is included 5 … extract the restaurant's `GeoCode` from the ReviewString 6 **Conditional** loop when match found with `GeoCode and >= 20000` (or break from FOR loop) 7 … selection statement to set Status to `"LOCATION FOUND "` **inside the loop** 8 Return `"NO REVIEW"` and `"LOCATION NOT FOUND"` 9 Return the Users review	

Question	Answer	Marks

5(b)	'Pseudocode' solution included here for development and clarification of mark scheme.
	Programming language example solutions appear in the Appendix.
	
	`FUNCTION AddReview(UserID: STRING) RETURNS BOOLEAN`
	
	`   DECLARE Index : INTEGER`
	`   DECLARE ReviewEntry : STRING`
	`   DECLARE Result : BOOLEAN`
	
	`   Index ← 1`
	`   Result ← FALSE`
	
	`   // get the result of CheckReview()`
	`   ReviewEntry ← CheckReview(UserID)`
	
	`   IF ReviewEntry <> "NO REVIEW" AND __`
	`      ReviewEntry <> "LOCATION NOT FOUND"`
	`      THEN`
	`         // get the next free index of Accepted array`
	`         WHILE Index <= 20000 AND Result = FALSE`
	`            IF Accepted[Index] = ""`
	`               THEN`
	`                  Accepted[Index] ← ReviewEntry`
	`                  Result ← TRUE`
	`            ENDIF`
	`            Index ← Index + 1`
	`         ENDWHILE`
	`   ENDIF`
	
	`   RETURN Result`
	
	`ENDFUNCTION`
	
	One mark for each of the following to Max 4
	
	1 Function header including parameter **and** return statement
	2 Call to function `CheckReview` with `UserID` as parameter (and assign to a variable)
	3 If a valid review
	4 … conditional loop with **two** conditions to find free index in `Accepted` array until `Counter = 20000`
	5 … store `ReviewEntry` in `Accepted` array
		4

Question	Answer	Marks

5(c)	
	```
	PROCEDURE BestRestaurants(SearchGeo : STRING)
	   DECLARE Index : INTEGER
	   DECLARE MatchFound : BOOLEAN
	   DECLARE Score : REAL // the extracted average score
	   DECLARE GeoCode : STRING // the extracted GeoCode
	   DECLARE ScoreLength: INTEGER // length of string to
	                                // extract score
	   Index ← 1
	   MatchFound ← FALSE

	   //linear search of ReviewScores for GeoCode and Score
	  // > 8.0
	   WHILE Index <= 20000 AND MatchFound = FALSE
	      GeoCode ← LEFT(ReviewScores[Index, 1], 7)
	      ScoreLength ← LENGTH(ReviewScores[Index, 1]) - 8;
	      Score ← STRING_TO_NUM(RIGHT(ReviewScores[Index,
	                                  1],ScoreLength)
	      IF GeoCode = SearchGeo AND Score > 8.0
	         THEN
	            MatchFound ← TRUE
	         ELSE
	            Index ← Index + 1
	      ENDIF
	   ENDWHILE

	   // output the review if found
	   IF MatchFound = TRUE
	      THEN
	         OUTPUT SearchGeo & " " & ReviewScores[Index, 2]
	   ENDIF

	ENDPROCEDURE
	```
	One mark for each of the following:
	
	1 extract `GeoCode` from `ReviewScores` **in any loop**
	2 extract the string representing `Score` from `ReviewScores` **in any loop**
	3 convert extracted string to a numeric value following any attempt at MP2 **in the loop**
	4 set `MatchFound` to `TRUE` if input parameter `GeoCode` is equal to extracted `GeoCode` **and** extracted score > 8.0 **in the loop**
	5 output parameter `SearchGeo` concatenated with its review text if `MatchFound` is `TRUE`
		5

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

Program Code Example Solutions

Q4(c): Visual Basic

```
    Sub Encrypt()
        Dim UnFileData, EnFiledata As String
        Dim Key, Counter As Integer
        Dim EnChar As Char

        Dim Sr As StreamReader = New StreamReader("DATA.txt")
        Dim Sw As StreamWriter = New StreamWriter("DATA-EN.txt ", True)
'append

        Do While Not Sr.EndOfStream
            UnFileData = Sr.ReadLine()
            Key = Convert.ToInt32(UnFileData.Substring(9, 2))
            EnFiledata = ""

            For Counter = 1 To UnFileData.Length
                EnChar = Convert.ToChar(Key +
                    Convert.ToByte(UnFileData.Substring(Counter, 1)))
                EnFiledata = EnFiledata & EnChar
            Next

            Sw.WriteLine(EnFiledata)
        Loop

        Sr.Close()
        Sw.Close()
    End Sub
```

Q4(c): Pascal

```
procedure Encrypt();
var
   UnFileData : String;
   EnFileData : String;
   EnFileName : String;
   Key, Counter : Integer;
   EnChar : String;

   DataFile : textfile;
   DataEnFile : textfile;
begin
   Assign(DataFile, 'DATA.txt ');
   reset(DataFile);
   EnFileName := 'DATA-EN.txt';

   while not eof(DataFile) do
   begin
      readln(DataFile, UnFiledata);
      Key := StrToInt(MidStr(UnFileData, 9, 2));
      EnFileData := '';

      for Counter := 1 to Length(UnFileData) do
      begin
         EnChar := Chr(Key+Ord(MidStr(UnFiledata, Counter, 1)[1]));
         EnFileData := EnFileData + EnChar;
      end;

      Assign(DataEnFile, EnFileName);
      append(DataEnFile);
      writeln(DataEnFile, EnFileData);
   end;

   Close(DataFile);
   Close(DataEnFile);
end;
```

Q4(c): Python

```
def Encrypt():
    #DECLARE UnFileData : STRING
    #DECLARE EnFileData : STRING
    #DECLARE Key, Counter : INTEGER
    #DECLARE EnChar : CHAR

    UnFilehandle = open("DATA.txt", "r")
    EnFilehandle = open("DATA-EN.txt", "a")

    UnFileData = UnFileHandle.readline()
    while UnFileData != "":
        Key = UnFiledata[9:11]
        EnFileData = ""

        for Counter in range(1, len(UnFileData)):
            EnChar = chr(Key + ord(UnFileData[Counter:1]))
            EnFileData = EnFileData + EnChar

        EnFilehandle.writeline(EnFileData)
        UnFileData = UnFileHandle.readline()

    UnFileHandle.close()
    EnFileHandle.close()
```

Q5(b): Visual Basic

```
    Function AddReview(UserID As String) As Boolean
        Dim Index As Integer
        Dim ReviewEntry As String
        Dim Result As Boolean

        Index = 1
        Result = False

        'get the result of CheckReview()
        ReviewEntry = CheckReview(UserID)

        If ReviewEntry <> "NO REVIEW" And
                ReviewEntry <> "LOCATION NOT FOUND" Then
            'get the next free index of Accepted array
            Do While Index <= 20000 And Result = False
                If Accepted(Index) = "" Then
                    Accepted(Index) = ReviewEntry
                    Result = True
                End If
                Index = Index + 1
            Loop
        End If

        Return Result

    End Function
```

Q5(b): Pascal

```
   function AddReview(UserID: string): boolean;
   var
      Index: integer;
      ReviewEntry: string;
      Result: boolean;
   begin
      Index := 1;
      Result := False;

      // get the result of CheckReview()
      ReviewEntry := CheckReview(UserID);

      if (ReviewEntry<>'NO REVIEW') and (ReviewEntry<>'LOCATION NOT FOUND')
then
      begin
         // get the next free index of Accepted array
         while (Index <= 20000) and (Result=False) do
         begin
            if Accepted[Index] = '' then
            begin
               Accepted[Index] := ReviewEntry;
               Result := True;
            end;
            Index := Index + 1;
         end;
      end;
      AddReview := Result;
   end;
```

Q5(b): Python

```
def AddReview(UserID):
    #DECLARE Index : INTEGER
    #DECLARE ReviewEntry : STRING
    #DECLARE Result : BOOLEAN

    Index = 1
    Result = False

    #get the result of CheckReview()
    ReviewEntry = CheckReview(UserID)

    if ReviewEntry != "NO REVIEW" and ReviewEntry != "LOCATION NOT FOUND":
        #get the next free index of Accepted array
        while Index <= 20000 and Result == False:
            if Accepted[Index] == "":
                Accepted[Index] = ReviewEntry
                Result = True
            Index += 1

    return Result
```