



Cambridge International AS & A Level

COMPUTER SCIENCE

9608/22

Paper 2 Fundamental Problem-Solving and Programming Skills

October/November 2021

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 2021 series for most Cambridge IGCSE™, Cambridge International A and AS Level components and some Cambridge O Level components.

This document consists of **19** printed pages.

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)	<p>One mark per bullet point to Max 4 e.g.</p> <p>Design stage activities:</p> <ul style="list-style-type: none"> • create / produce / define identifier table // decide on identifiers • create / produce / define data structures // data types • create / produce / define file structures • create / produce / define test plan / strategy • create pseudocode • create flowchart etc. • identify inputs/outputs/processes • decomposing the problem into sub-problems • choose a suitable programming language <p>Coding stage activities:</p> <ul style="list-style-type: none"> • write program code // coding the algorithms (from design) 'write code' on its own is NE • define data structures • use a translator to check and run the code • initial debugging // testing • any example of actions performed when programming 	4										
1(b)(i)	<p>One mark per bullet point to Max 2</p> <ul style="list-style-type: none"> • names are not meaningful (or equivalent) // name does not reflect the identifier's use • makes the program/variables more difficult to understand // difficult for non-technical/other person to understand the program/variables • makes the program more difficult to debug / modify / test 	2										
1(b)(ii)	<p>One mark for each correct row.</p> <table border="1" data-bbox="304 1218 1254 1541"> <thead> <tr> <th data-bbox="304 1218 411 1283">Line</th> <th data-bbox="411 1218 1254 1283">Appropriate identifier name</th> </tr> </thead> <tbody> <tr> <td data-bbox="304 1283 411 1348">102</td> <td data-bbox="411 1283 1254 1348">Price / ItemPrice / StockPrice</td> </tr> <tr> <td data-bbox="304 1348 411 1413">103</td> <td data-bbox="411 1348 1254 1413">ExpiryDate / ItemExpiryDate / EndDate</td> </tr> <tr> <td data-bbox="304 1413 411 1478">105</td> <td data-bbox="411 1413 1254 1478">LowStockValue / LowValue / LowStock</td> </tr> <tr> <td data-bbox="304 1478 411 1541">106</td> <td data-bbox="411 1478 1254 1541">IsOutOfStock / IsInStock</td> </tr> </tbody> </table>	Line	Appropriate identifier name	102	Price / ItemPrice / StockPrice	103	ExpiryDate / ItemExpiryDate / EndDate	105	LowStockValue / LowValue / LowStock	106	IsOutOfStock / IsInStock	4
Line	Appropriate identifier name											
102	Price / ItemPrice / StockPrice											
103	ExpiryDate / ItemExpiryDate / EndDate											
105	LowStockValue / LowValue / LowStock											
106	IsOutOfStock / IsInStock											
1(b)(iii)	<p>One mark each</p> <ul style="list-style-type: none"> • (Constant) declaration with appropriate identifier for program version (as String) • Storing correct value in the variable (equals or arrow) <p>e.g. <code>CONSTANT ProgramVersion = "ver1.5.8"</code></p>	2										
1(c)	<p>One mark each to Max 2</p> <ul style="list-style-type: none"> • wider range of character can be represented • symbols from (more) languages can be represented • pictograms / emoticons can be represented 	2										

Question	Answer	Marks															
2(a)	One mark each of the following to max 5: <ul style="list-style-type: none"> • Input the membership number • Open the file LOYALTY.txt for read and close the file • Loop until end of file (and membership number not found) • ...read each line from the file • Extract stored membership number and points • If number of points is 10 or more, return/output free cake • If membership number not found, return/output appropriate value/message • If number of points is below 10, return / output no free cake • Subtract 10 points from their points and store 	5															
2(b)	One mark for each correct row. <table border="1" data-bbox="304 730 1321 1585" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th data-bbox="304 730 624 831">Description of expression</th> <th data-bbox="624 730 1158 831">Pseudocode expression</th> <th data-bbox="1158 730 1321 831">Evaluates to</th> </tr> </thead> <tbody> <tr> <td data-bbox="304 831 624 1003">Evaluates to TRUE if DayOfMonth is within the first seven days of the month</td> <td data-bbox="624 831 1158 1003"> <pre>STRING_TO_NUM(DayOfMonth) <= 7 // DayOfMonth <= "7"</pre> </td> <td data-bbox="1158 831 1321 1003" style="text-align: center;">FALSE</td> </tr> <tr> <td data-bbox="304 1003 624 1245">Concatenates the second and third letters of Firstname and concatenate with the last three letters of Lastname</td> <td data-bbox="624 1003 1158 1245"> <pre>MID(Firstname, 2, 2) & RIGHT(Lastname, 3) // MID(Firstname, 2, 2) & MID(Lastname, LENGTH(Lastname) - 3, 3)</pre> </td> <td data-bbox="1158 1003 1321 1245" style="text-align: center;">"eason"</td> </tr> <tr> <td data-bbox="304 1245 624 1384">Evaluates to TRUE if DOB contains eight characters</td> <td data-bbox="624 1245 1158 1384"> <pre>LENGTH(DOB) = 8</pre> </td> <td data-bbox="1158 1245 1321 1384" style="text-align: center;">TRUE</td> </tr> <tr> <td data-bbox="304 1384 624 1585">Evaluates to TRUE if the customer is a member and has enough points for a free slice of cake</td> <td data-bbox="624 1384 1158 1585"> <pre>IsMember AND Points >= 10 IsMember = TRUE AND Points >= 10</pre> </td> <td data-bbox="1158 1384 1321 1585" style="text-align: center;">TRUE</td> </tr> </tbody> </table>	Description of expression	Pseudocode expression	Evaluates to	Evaluates to TRUE if DayOfMonth is within the first seven days of the month	<pre>STRING_TO_NUM(DayOfMonth) <= 7 // DayOfMonth <= "7"</pre>	FALSE	Concatenates the second and third letters of Firstname and concatenate with the last three letters of Lastname	<pre>MID(Firstname, 2, 2) & RIGHT(Lastname, 3) // MID(Firstname, 2, 2) & MID(Lastname, LENGTH(Lastname) - 3, 3)</pre>	"eason"	Evaluates to TRUE if DOB contains eight characters	<pre>LENGTH(DOB) = 8</pre>	TRUE	Evaluates to TRUE if the customer is a member and has enough points for a free slice of cake	<pre>IsMember AND Points >= 10 IsMember = TRUE AND Points >= 10</pre>	TRUE	4
Description of expression	Pseudocode expression	Evaluates to															
Evaluates to TRUE if DayOfMonth is within the first seven days of the month	<pre>STRING_TO_NUM(DayOfMonth) <= 7 // DayOfMonth <= "7"</pre>	FALSE															
Concatenates the second and third letters of Firstname and concatenate with the last three letters of Lastname	<pre>MID(Firstname, 2, 2) & RIGHT(Lastname, 3) // MID(Firstname, 2, 2) & MID(Lastname, LENGTH(Lastname) - 3, 3)</pre>	"eason"															
Evaluates to TRUE if DOB contains eight characters	<pre>LENGTH(DOB) = 8</pre>	TRUE															
Evaluates to TRUE if the customer is a member and has enough points for a free slice of cake	<pre>IsMember AND Points >= 10 IsMember = TRUE AND Points >= 10</pre>	TRUE															

Question	Answer	Marks
2(c)	<p>One mark for each bullet point</p> <ul style="list-style-type: none"> • CASE statement has correct structure (all key words and format) • Correct assignment/identification of all 3 file types in case statement • Assignment/return of "Unknown" as default • Return FileType instead of FileExt // Correct return value for all file types and Unknown <pre> FUNCTION GetFileType(Filename : STRING) RETURNS STRING DECLARE FileExt : STRING DECLARE FileType: STRING FileExt ← RIGHT(Filename, 3) CASE OF FileExt 'rtf' : FileType ← "Rich text format" 'csv' : FileType ← "Comma separated values" 'txt' : FileType ← "Text" OTHERWISE : FileType ← "Unknown" ENDCASE RETURN FileType ENDFUNCTION </pre>	4

Question	Answer	Marks
3(a)	<p>One mark each to max 3</p> <ul style="list-style-type: none"> • break the problem/algorithm (not program / code) into smaller steps / parts/ subproblems • ... repeatedly only if MP1 given • ... until all subproblems small/detailed enough to solve • ... to identify program modules // to identify repeated elements // for modular programming • ... to identify subroutines 	3
3(b)	<p>1 mark for definition of term e.g.</p> <ul style="list-style-type: none"> • apply current knowledge to an unfamiliar scenario <p>1 mark for how the skills are used in program development e.g.</p> <ul style="list-style-type: none"> • use current knowledge of a familiar programming language in a new language 	2

Question	Answer	Marks
4(a)	<p>One mark for each bullet point</p> <ul style="list-style-type: none">• Procedure heading and ending• Loop from 1 to 20 // Loop from 0 to 19 Allow NEXT for ENDFOR• Assign -1 to array at loop counter• All logic correct, does not override Flower array, all 20 elements assigned -1 <p>e.g.</p> <pre>PROCEDURE InitialiseArray() DECLARE Index : INTEGER FOR Index ← 1 TO 20 Flower[Index] ← -1 ENDFOR ENDPROCEDURE</pre>	4

Question	Answer	Marks
4(b)	<p>One mark for each bullet point:</p> <ul style="list-style-type: none"> • Declare variable to store first flower visited and initialise to 10 // Store flower 10 in first array index • Loop until all 20 flowers are stored • Generate random number between 1 and 20 in the loop ... • ... Check if array at random number is -1 // Check if random number is already in flower array • ... If it is not -1 / already in array, generate another random number • ... repeatedly until -1 / not in array • ... otherwise assign Flower at previous flower to random number // otherwise assign random number to next array element and increment counter <p>'Pseudocode' solution included here for development and clarification of mark scheme. Programming language example solutions appear at the end.</p> <p>Example 1:</p> <pre> PROCEDURE RandomPath() DECLARE New : INTEGER DECLARE Previous : INTEGER New ← 10 Previous ← 10 FOR X ← 0 TO 19 WHILE NOT(Flower[New] = -1) New ← INT(RAND(20)+1) Flower[Previous] ← New ENDWHILE ENDFOR ENDPROCEDURE </pre> <p>Example 2:</p> <pre> PROCEDURE RandomPath() DECLARE Current : INTEGER DECLARE Counter: INTEGER DECLARE Next : INTEGER Current ← 10 Counter ← 1 WHILE Counter <= 20 Next ← INT(RAND(20)+1) IF Flower[Next] = -1 THEN Flower[Current] ← Next Current ← Next Counter ← Counter + 1 ENDIF ENDWHILE ENDPROCEDURE </pre>	7

Question	Answer	Marks
4(c)(i)	1 mark for error type: <ul style="list-style-type: none"> Logic // run-time 1 mark for method of detection e.g. <ul style="list-style-type: none"> Testing the program with a range of data Whitebox testing Blackbox testing 	2
4(c)(ii)	1 mark each to max 2 e.g. <ul style="list-style-type: none"> Program crashes Error as number is out of array bounds // number below 1 will produce error // number above 20 will produce error Not all elements will be accessed/used Error if decimal generated Error if negative number generated Infinite loop may be generated 	2

Question	Answer	Marks
5(a)	1 mark each <ul style="list-style-type: none"> Start and end Start/begin End/stop Procedure/end procedure For each circled area (Max 4) <pre> graph TD START([START]) --> SET1[SET OutYear TO NUM_TO_STRING(Year)] SET1 --> IS_TRUE{Is UseSuffix TRUE?} IS_TRUE -- YES --> IS_NEG{Is Year < 0?} IS_TRUE -- NO --> OUTPUT[/OUTPUT OutYear/] IS_NEG -- YES --> SET2[SET OutYear TO OutYear & SUFFIX1] IS_NEG -- NO --> SET3[SET OutYear TO OutYear & SUFFIX2] SET2 --> OUTPUT SET3 --> OUTPUT OUTPUT --> END([END]) </pre> <p>1 mark for START and END. 1 mark for each individual zone as shown.</p>	5

Question	Answer	Marks
5(b)	1 mark for values, 1 mark for result from: <ul style="list-style-type: none"> • Year >= 0 UseSuffix = TRUE • Matching Year value with "CE" • Year >= 0 UseSuffix = FALSE • Matching Year value • Year < 0 UseSuffix = TRUE • Matching Year value with "BCE" • Year < 0 UseSuffix = FALSE • Matching Year value 	6
5(c)(i)	One mark each to max 2. <ul style="list-style-type: none"> • code is unknown/not considered • data is chosen to test boundaries // test with normal, extreme and erroneous • compare expected output with actual output // testing whether the inputs produce the correct/expected outputs • testing if the program meets the requirements 	2
5(c)(ii)	One mark each to max 2. <ul style="list-style-type: none"> • data is chosen to test algorithm/code • tests every path in the code • test each line of code/structure • internal structure is being tested • tester can view code 	2

Question	Answer	Marks
6(a)	<p>One mark each to max 6:</p> <ul style="list-style-type: none"> • Function declared taking the string to search for as a parameter, string return (and array) • (Declare and) initialise a variable (0) to count number times code occurs • Loop through 20 000 elements ... // looping until unused element • ... IF Left 7 from GeoCodeLog[loop counter] = parameter ... • ... if true, increment number times occurs • ... if true, RIGHT 10 from GeoCodeLog[loop counter] ... • ... if true compare to last date and replace if after // store in variable for last date • Return concatenated number times & " " & last date ... • ... converting number to string <p>Example 1:</p> <pre> FUNCTION SearchLog(SearchGeoCode : STRING) RETURNS STRING DECLARE AccessCount : INTEGER DECLARE LatestDate : STRING DECLARE DateAccess : DATE LatestDate ← "01/01/1500" FOR Index ← 0 TO 19999 IF LEFT(GeoCodeLog[Index], 7) THEN AccessCount ← AccessCount + 1 DateAccess ← (RIGHT(GeoCodeLog[Index], 10)).TODATE IF DateAccess > LatestDate THEN LatestDate ← DateAccess ENDIF ENDIF ENDFOR RETURN NUM_TO_STRING(AccessCount) & " " & DATE_TO_STRING(LatestDate) ENDFUNCTION </pre>	6

Question	Answer	Marks
6(a)	<p>Example 2:</p> <pre> FUNCTION SearchLog(SearchGeoCode : STRING) RETURNS STRING DECLARE Index : INTEGER DECLARE AccessCount : INTEGER DECLARE LogGeoCode : STRING DECLARE LastDate : STRING DECLARE CountDateLine : STRING CONSTANT SPACE = ' ' Index ← 1 AccessCount ← 0 LogGeoCode ← "" LastDate ← "" WHILE Index <= 20000 AND LogGeoCode <> "AAAA+0A" LogGeoCode ← LEFT(GeoCodeLog[Index], 7) IF LogGeoCode = SearchGeoCode THEN AccessCount ← AccessCount + 1 LastDate ← RIGHT(GeoCodeLog[Index], 10) ENDIF Index ← Index + 1 ENDWHILE CountDateLine ← NUM_TO_STRING(AccessCount) & SPACE & LastDate RETURN CountDateLine ENDFUNCTION </pre>	

Question	Answer	Marks
6(b)	<p>1 mark each to max 7</p> <ul style="list-style-type: none"> • Procedure header • Open file "Locations.txt" in WRITE mode and close the file • Loop through 20 000 elements // looping until unused element ... • ... if GeoCodeData[loop counter] = "AAAA+0A" loop again/exit • ... if not empty call SearchLog() with LEFT(GeoCodeData[DataIndex], 7) ... FT invalid/missing if • ... use/store returned value • ... replace the space with a # • ... concatenate GeoCodeData[loop counter] & "#" & returned data inside the loop ... • ... writing this value to file inside the loop <p>'Pseudocode' solution included here for development and clarification of mark scheme.</p> <p>Programming language example solutions appear at the end.</p> <p>Example 1:</p> <pre> PROCEDURE ExtractArrays() DECLARE dataIndex : INTEGER DECLARE ShortCode, CountAndDate, LocationLine, TempCode : STRING CONSTANT FILENAME = "Locations.txt" dataIndex ← 1 ShortCode ← "" TempCode ← "" OPENFILE FILENAME FOR WRITE FOR Count ← 0 TO 19999 ShortCode ← LEFT(GeoCodeData[dataIndex], 7) IF NOT(ShortCode = "AAAA+0A") THEN CountAndDate ← SearchLog(ShortCode) FOR Index ← 0 TO LENGTH(CountAndDate)-1 IF NOT(MID(CountAndDate, Index, 1) = " ") THEN TempCode = TempCode & MID(CountAndDate, Index, 1) ELSE TempCode = TempCode & "#" ENDIF ENDFOR LocationLine ← GeoCodeData[dataIndex] & "#" & TempCode WRITEFILE(FILENAME, LocationLine) </pre>	7

Question	Answer	Marks
6(b)	<pre> ENDIF ENDFOR CLOSEFILE FILENAME ENDPROCEDURE Example 2: PROCEDURE ExtractArrays() DECLARE dataIndex : INTEGER DECLARE ShortCode, CountAndDate, LocationLine : STRING CONSTANT HASH = '#' CONSTANT FILENAME = "Locations.txt" dataIndex ← 1 ShortCode ← "" OPENFILE FILENAME FOR WRITE WHILE dataIndex ≤ 20000 AND ShortCode <> "AAAA+0A" ShortCode ← LEFT(GeoCodeData[dataIndex], 7) IF ShortCode <> "AAAA+0A" THEN CountAndDate ← SearchLog(ShortCode) LocationLine ← GeoCodeData[dataIndex] & HASH & CountAndDate WRITEFILE (FILENAME, LocationLine) ENDIF dataIndex ← dataIndex + 1 ENDWHILE CLOSEFILE FILENAME ENDPROCEDURE </pre>	

Program Code Example Solutions**Q4 (b): Visual Basic**

```
Sub RandomPath()  
    ' alternative solution  
    Dim Index, Current, NextFlower As Integer  
    Dim AllFlowersVisited As Boolean  
    Dim RandomFlower As New Random  
  
    Current = 10  
    AllFlowersVisited = False  
  
    Do While AllFlowersVisited = False  
        NextFlower = RandomFlower.Next(1, 20)  
  
        If Flower(NextFlower) = 0 Then  
            If Flower(Current) = 0 Then  
                Flower(Current) = NextFlower  
            End If  
            Current = NextFlower  
        End If  
        AllFlowersVisited = True  
        For Index = 1 To 20  
            If Flower(Index) = 0 Then  
                AllFlowersVisited = False  
            End If  
        Next  
    Loop  
End Sub
```

Q4 (b): Pascal

```
procedure RandomPath();
// alternative solution
var
  Index, Current, Next: integer;
  AllFlowersVisited : boolean;

begin
  Current := 10;
  AllFlowersVisited := False;

  while (AllFlowersVisited = False) do
  begin
    Randomize;
    Next := RandomRange(1, 21);

    if (Flower[Next] = 0) then
    begin
      if (Flower[Current] = 0) then Flower[Current] := Next;
      Current := Next;
    end;
    AllFlowersVisited := True;
    for Index := 1 to 20 do
    begin
      if Flower[Index] = 0 then
        AllFlowersVisited := False;
      end;
    end;
  end;
end;
```

Q4 (b): Python

```
from random import seed
from random import randint

def RandomPath():
    # alternative solution
    #DECLARE Index : INTEGER
    #DECLARE Current : INTEGER
    #DECLARE Next : INTEGER
    #DECLARE AllFlowersVisited : BOOLEAN

    Current = 10 #start at flower 10 in the field
    AllFlowersVisited = False

    while AllFlowersVisited == False:
        #loop until all flowers visited
        Next = randint(20)
        if Flower[Next] == 0:
            if Flower[Current] == 0: #do not revisit a flower
                Flower[Current] = Next
            Current = Next

    AllFlowersVisited = True #assume all visited unless...
    for Index in range(1, 20):
        if Flower[Index] == 0:
            AllFlowersVisited = False #... proved otherwise
```


Q6 (b): Visual Basic

```
Sub ExtractArrays()  
    Dim DataIndex As Integer  
    Dim ShortCode, CountAndDate, LocationLine, CountHashDate As String  
    Dim Sr As StreamWriter = New StreamWriter("Locations")  
  
    Const HASH = "#"  
  
    DataIndex = 1  
    ShortCode = ""  
    CountHashDate = ""  
  
    Do While DataIndex <= 20000 And ShortCode <> "AAAA+0A"  
        CountHashDate = ""  
        ShortCode = GeoCodeData(DataIndex).SubString(0, 7)  
        If ShortCode <> "AAAA+0A" Then  
            CountAndDate = SearchLog(ShortCode)  
            For Index = 0 To CountAndDate.Length()-1  
                If (MID(CountAndDate, Index, 1) = " ") Then  
                    CountHashDate = CountHashDate & "#"  
                Else  
                    CountHashDate = CountHashDate & MID(CountAndDate, Index, 1) = ""  
                End If  
            Next Index  
            LocationLine = GeoCodeData(DataIndex) & HASH & CountHashDate  
            Sr.WriteLine(LocationLine)  
        End If  
        DataIndex = DataIndex + 1  
    Loop  
    Sr.Close()  
End Sub
```

Q6 (b): Pascal

```
procedure ExtractArrays();
var
  DataIndex: integer;
  ShortCode, CountAndDate, LocationLine, CountHashDate: string;
  Locations: textfile;
const
  HASH = '#';
begin
  DataIndex := 1;
  ShortCode := '';

  assign(Locations, 'M:\Locations');
  rewrite(Locations);

  while (DataIndex <= 20000) and (ShortCode <> 'AAAA+0A') do
  begin
    CountHashDate := "";
    ShortCode := LeftStr(GeoCodeData[DataIndex], 7);
    if ShortCode <> 'AAAA+0A' then
    begin
      CountAndDate := SearchLog(ShortCode);
      for Index := 0 to Length(CountAndDate)-1 do
      begin
        if CountAndDate[Index] = " " then
          CountHashDate := CountHashDate + "#"
        else
          CountHashDate := CountHashDate + CountAndDate[Index];
        end;

        LocationLine := GeoCodeData[DataIndex] + HASH + CountHashDate;
        writeln(Locations, LocationLine);
      end;
      DataIndex := DataIndex + 1;
    end;
  close(Locations);
end;
```

Q6 (b): Python

```
def ExtractArrays():
    #DECLARE dataIndex : INTEGER
    #DECLARE ShortCode, CountAndDate, LocationLine, CountHashDate : STRING

    HASH = "#"
    FILENAME = "Locations"

    dataIndex = 1
    ShortCode = ""

    filehandle = open(FILENAME, 'w')

    while dataIndex <= 20000 and ShortCode <> "AAAA+0A":
        ShortCode = GeoCodedata[dataIndex][:7]
        CountHashDate = ""
        if ShortCode <> "AAAA+0A":
            CountAndDate = SearchLog(ShortCode)
            for Index in range(0, Len(CountAndDate)):
                if (CountAndDate[Index] == " "):
                    CountHashDate += "#"
                else:
                    CountHashDate += CountAndDate[Index]
            LocationLine = GeoCodeData[dataIndex] + HASH + CountHashDate
            filehandle.write(LocationLine)
        dataIndex += 1

    filehandle.close()
```