Chapter 11

End of Chapter Exercises

1. *Have a closer look at statements #9.3.1 and #9.4.1 in Program 11.5. Both statements write `name` and salary to `reportFile` — the only difference is the string they append to the end of the line. Give two alternative solutions to the file writing section.*

First alternative file writing section:

```
WHILE NOT (EndOfFile (salaryFile))
   Get (salaryFile, REFERENCE:name, REFERENCE:salary) ;
   Display (name, ':', salary↲) ;
   Write (reportFile, name, salary) ;
   IF (salary > 25000)
      Write (reportFile, 'High'↲) ;
   ELSE
      Write (reportFile, 'Normal'↲) ;
   ENDIF
ENDWHILE
```

Second alternative file writing section:

```
string:salarySize ;
WHILE NOT (EndOfFile (salaryFile))
   Get (salaryFile, REFERENCE:name, REFERENCE:salary) ;
   Display (name, ':', salary↲) ;
   IF (salary > 25000)
      salarySize ← 'High' ;
   ELSE
      salarySize ← 'Normal' ;
   ENDIF
   Write (reportFile, name, salary, salarySize↲) ;
ENDWHILE
```

2. *What is wrong with the algorithm fragment below?*

```
integer:myNumber ;
Display ('Please enter an integer value: ') ;
Get (keyboard, myNumber) ;
```

`Get` needs to take a reference parameter:

```
Get (keyboard, REFERENCE:myNumber) ;
```

3. *In the exercises for Chapter 6 you were asked to write an algorithm to display a multiplication table. Amend your algorithm so that the table is written to a text file instead of the screen. The table to be calculated is stored as a single integer value in a text file* `table.txt` *which your program needs to read.*

```
file:table ,
     outTable;
table ← 'table.txt' ;
outTable ← 'out.txt' ;
Open (table) ;
Open (outTable) ;
Get (table, REFERENCE:timesTable) ;
FOR counter GOES FROM 1 TO timesTable
   product ← counter × 12 ;
   Write (outTable, counter ' × ' 12 ' = ' product↵) ;
ENDFOR
Close (table) ;
Close (outTable) ;
```

4. *The algorithm below is intended to read from a file* `distances.txt` *which contains a number of real values representing distances in nautical miles between Stocksfield and a number of other cities. Each line contains a string holding the city name followed by a real value holding the distance to that city. There is no terminating record. Table 11.2 shows the cities and distances in nautical miles that are held in* `distances.txt` *as well as showing what the calculated distance in km should be.*

City	Distance from Stocksfield	
	Nautical Miles	Km
Wylam	3.21	5.94
Newcastle	10.75	19.91
Edinburgh	75.02	138.93
Dublin	179.98	333.32
Cardiff	211.07	390.91
London	215.88	399.81
Paris	396.35	734.04
Berlin	561.44	1039.78
Jerusalem	2093.40	3876.98
Ottawa	2755.48	5103.14
Washington DC	3062.38	5671.52
Canberra	9123.53	16896.77
Wellington	9971.13	18466.53

The algorithm converts these distances to kilometres (1 nautical mile = exactly 1,852 metres) and displays the converted distances on the monitor screen. There are 3 errors in the algorithm. State what they are and correct them.

A new statement 9 is needed to open the distances file. Statement 10.1 needs reference parameters. Statement 10.2 needs to divide by 1000 to convert from metres to km. Changes shown in bold & bold line number.

```
1.  integer:metresPerNM IS 1852 ;
2.  string:city ;
3.  real:distance,
4.       distanceInKM ;
5.  file:distancesFile ;
6.  distancesFile ← distances.txt' ;
7.  Display ('Distances of cities from Stocksfield (km)↲') ;
8.  Display ('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~↲') ;
9.  Open (distancesFile) ;
10. WHILE NOT EndOfFile (distancesFile)
       10.1.  Get (distancesFile, REFERENCE:city,
                   REFERENCE:distance) ;
       10.2.  distanceInKM ← distance × metresPerNM ÷ 1000 ;
       10.3.  Display (city, '=', distanceinKM ,'km'↲) ;
    ENDWHILE
11. Close (distancesFile) ;
```

5. *Having found and corrected the errors in the previous exercise, now amend the algorithm so that it deals with a differently formatted distance.txt in which a line containing a city name of `'Stocksfield'` and a distance of `0.0` marks the end of the data.*

```
1.  integer:metresPerNM IS 1852 ;
2.  string:city ;
3.  real:distance,
4.       distanceInKM ;
5.  file:distancesFile ;
6.  distancesFile ← distances.txt' ;
7.  Display ('Distances of cities from Stocksfield (km)↲') ;
8.  Display ('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~↲') ;
9.  Open (distancesFile) ;
10. Get (distancesFile, REFERENCE:city, REFERENCE:distance) ;
10. WHILE (city ≠ Stocksfield) OR (distance ≠ 0.0)
       10.2.  distanceInKM ← distance × metresPerNM ÷ 1000 ;
```

```
            10.3.  Display (city, '=', distanceinKM ,'km'↲) ;
            10.4.  Get (distancesFile, REFERENCE:city,
                        REFERENCE:distance) ;
        ENDWHILE
    11. Close (distancesFile) ;
```

Projects

StockSnackz Vending Machine

Stocksfield Fire Service

In the main program we can declare the file:

```
file:logFile ;
logFile ← 'log.txt' ;
Open (logFile) ;
```

Now we can add the file outputs to the function.

```
FUNCTION ProcessEAC (string:EAC) RETURNS Boolean
   Boolean:valid ;
   valid ← True ;
   IF (EAC [0] = '1')
      Display ('Use coarse spray↲') ;
      Write (logFile, 'Use coarse spray↲') ;
   ELSE IF (EAC [0] = '2')
      Display ('Use fine spray↲') ;
      Write (logFile, 'Use fine spray↲') ;
```

... and so on.

```
ENDFUNCTION
```

More properly, we should really pass the log file as a parameter to the function.

Puzzle World: Roman numerals & chronograms

No solutions provided as they are highly dependent on how you have structured your own solutions over the previous chapters.

Pangrams: holoalphabetic sentences

No solutions provided as they are highly dependent on how you have structured your own solutions over the previous chapters.

Online bookstore: ISBNs

No solutions provided as they are highly dependent on how you have structured your own solutions over the previous chapters.