(a) Emma has written an algorithm in pseudocode. This is shown in the diagram below. The code contains a subroutine.

- Line numbers are included but are not part of the algorithm

```
 1  SUBROUTINE functionA(aTemperature)
 2      temp <- (aTemperature * 9)/5 + 32
 3      RETURN temp
 4  ENDSUBROUTINE
 5
 6  OUTPUT "Enter the current temperature"
 7  theTemp <- USERINPUT
 8
 9  OUTPUT "Enter F for Fahrenheit or C for Celsius"
10  units <- USERINPUT
11
12  IF units = "F" THEN
13      theTemp <- functionA(theTemp)
14      OUTPUT "Temperature in Celsius is " + theTemp
15  ENDIF
```

(i) Define the term algorithm.

[2 marks]

a set of instructions [1] to complete a task [1]

Do not allow: computer program or reference to an algorithm as only a program

(ii) units is a variable used in the main program in Emma's algorithm.

Write down the line number where units is declared

[1 mark]

10

(iii) Define the term variable as it is used in computer programming.

[2 marks]

A named area of memory [1] where data can be stored [1]

(iv) temp is is a local variable used in the algorithm. State the data type used for temp.

[1 mark]

any of: real [1] float [1] real number [1] floating point number [1]

(v) Explain what the term local variable means.

[2 marks]

a variable used in a subroutine/function [1] that can not be used outside of the subroutine [1]

a variable that only exists [1] within a subroutine [1]

(vi) Write down one line number where user input takes place

[1 mark]

7 or 10

(vii) Write down the line number where selection is first used in the algorithm

[1 mark]

12

(viii) Write down the line numbers which make up the subroutine in the algorithm

[1 mark]

1 – 4 (accept 1 – 3 but be grumpy about it)

(ix) Explain why the name given to the subroutine in the code may be confusing.

[2 marks]

functionA gives no idea what the subroutine is used for [1] and so may confuse other programmers [1]

the name does not make it clear what the subroutine does [1]

do not all it is confusing (in question)

(b) Functions are an example of decomposition.

(i) Explain what the term decomposition means in computer programming.

[1 mark]

to break a problem down into parts [1]

(ii) Explain the advantages of using decomposition when writing programs.

[6 marks]

Level 1: 1-2 marks. Simple points without any development. All lists get L1 only

e.g. work can be done by different programmers; easier to test

Level 2: 3-4 marks. Points which show some basic development. Any L2 point will get the answer into L2.

e.g. subroutines can be tested once they are written to make sure they work properly

Level 3: 5-6 marks. At least 4 points made, mostly showing some development

e.g. code can be reused from other projects meaning that it will be quicker and easier to code the problem

e.g. different programmers can work on individual problems so that the work gets done more efficiently