# Worksheet 2 Structured programming

**Task 1**

1. The following program is written as it would have been before the days of structured programming.

It is designed to allow a user to input the times taken, to the nearest minute, by different people to complete a certain task. The program outputs the number of people who took

* more than 60 minutes
* between 30 and 59 minutes
* less than 30 minutes

slow = 0

medium = 0

fast = 0

INPUTDATA

timeTaken = USERINPUT

IF timeTaken = 0 GOTO PRINT

IF timeTaken < 30 GOTO UNDER30

IF timeTaken <60 GOTO UNDER60

slow = slow + 1

GOTO INPUTDATA

UNDER30

fast = fast + 1

GOTO INPUTDATA

UNDER60

medium = medium + 1

GOTO INPUTDATA

PRINT

OUTPUT fast, medium, slow

(i) Rewrite the programming using structured programming techniques.

(ii) Flowcharts were invented in the days of GOTO statements. That is why they are not well-suited to representing iteration and selection structures. Can you draw a flowchart of the unstructured program?

(iii) Which version of the program is

* quicker to write?
* easier to understand?
* less likely to contain errors?

(iv) Another feature of some early programming languages was that no identifier (e.g. variable name or label) could be more than 6 characters. How would this affect program readability, ease of debugging and maintenance?

2. (a) MOD is an arithmetic operator which returns the remainder from integer division.

e.g. x = 27 MOD 4 will put the value 3 in x.

DIV returns the integer result of the division.

e.g. y = 27 DIV 4 will put the value 6 in y.

Write pseudocode statements to allow the user to input a 3-digit number, and then output the individual digits in the number.

e.g. If the user enters 465, the output should be “The digits are 4 6 5”

(b) Devise a pseudocode algorithm which generates and prints all 3-digit numbers that equal the sum of the cubes of their individual digits.

e.g. 153 satisfies this condition because 153 = 13 + 53 + 33

(In pseudocode, express this as 1\*\*3 + 5\*\*3 + 3\*\*3)

**Task 2**

3. A hierarchy chart can be compared to an upside-down tree, with the root at the top and branches and leaves spreading downwards.

The “leaves” are the lowest level modules and all or most of the detailed program code will be in the “leaves”.

In the hierarchy chart below:

(a) Which are the Level 1 modules?

(b) Which are the Level 2 modules?

(c) Which are the Level 3 modules?

(d) Write down the order in which the modules are executed.

**6**

**1**

**3**

**2**

**4**

**7**

**9**

**8**

**11**

**10**

**5**

4. The following pseudocode program is designed to allow the user to input a series of three numbers and for each set of numbers, find and output the maximum. The maximum is then added to a total. When the user enters 000 for the three numbers, the average of all the maximums is calculated and output.

SUB initialise

OUTPUT “This program finds the maximums of sets of three numbers.

Enter three zeroes when all numbers entered.

Program then calculates and outputs the average of the maximums”

total = 0

n = 0

ENDSUB

SUB promptForNumbers

OUTPUT"Please enter first number "

num1 = USERINPUT

OUTPUT ("Please enter second number "

num2 = USERINPUT

OUTPUT "Please enter third number "

num3 = USERINPUT

ENDSUB

SUB findMax

maxnum = num1

IF num2>maxnum THEN

maxnum = num2

ELSE

IF num3>maxnum THEN

maxnum = num3

ENDIF

OUTPUT "Max of the three numbers is is ",maxnum

ENDSUB

SUB performCalculations

total = total + maxnum

n=n+1

ENDSUB

SUB processData

promptForNumbers

WHILE num1<>0 and num2<>0 and num3<>0

findMax

performCalculations

promptForNumbers

ENDWHILE

ENDSUB

SUB calculateAverage

average = total/n

OUTPUT "Average of maximums is ",average

ENDSUB

#Main program starts here

initialise

processData

calculateAverage

Draw a hierarchy chart representing this program. Show the different levels, i.e. Level 1 modules, Level 2 modules etc.