Cambridge International AS & A Level

COMPUTER SCIENCE **9608/22**

Paper 2 Fundamental Problem-solving and Programming Skills

May/June 2020

SOLVED PRE-RELEASE MATERIAL by SIR MAJID TAHIR

TASK 1 – Structure charts

Describe a processing activity that can be represented by one main task with two or more sub-tasks. The activity can relate to any scenario, but should include aspects of selection and iteration.

Activity examples may be taken from different areas, such as:

- school or college
- factory or workplace
- clubs or hobbies.

TASK 1.1
Consider how a problem is decomposed by splitting it into smaller parts.
Discuss the advantages of this approach.

ANSWER
Activity examples is taken from different areas, such as: **School or Collage**

Decomposition/Step wise refinement

- To make it easier to solve a bigger problem, we break the problem into smaller steps.
- These might need breaking down further until the steps are small enough to solve
- For a solution to a problem to be programmable, we need to break down the steps of the solution into the basic constructs of sequence, assignment, selection, repetition, input and output.
- We can use a method called stepwise refinement to break down the steps of our outline solution into smaller steps until it is detailed enough.
- A solution is to decompose the problem into sub-tasks.
- **Each sub-task can be considered as a 'module' that is refined separately.**
- **Modules are procedures and functions.**
- **A procedure groups together a number of steps and gives them a name (an identifier).**
- **We can use this identifier when we want to refer to this group of steps. When we want to perform the steps in a procedure we call the procedure by its name**

Advantages:

- This structured approach works for the development of both large and small computer systems.
- When large computer systems are being developed this means that several programmers can work independently to develop and test different subsystems/modules
- Multiple modules can be programmed at the same time.
- This reduces the development and testing time.

- Debugging the program is easier as separate modules can be checked instead of whole program
- Modules can be tested separately and then combined in the complete program.
- More robust code.

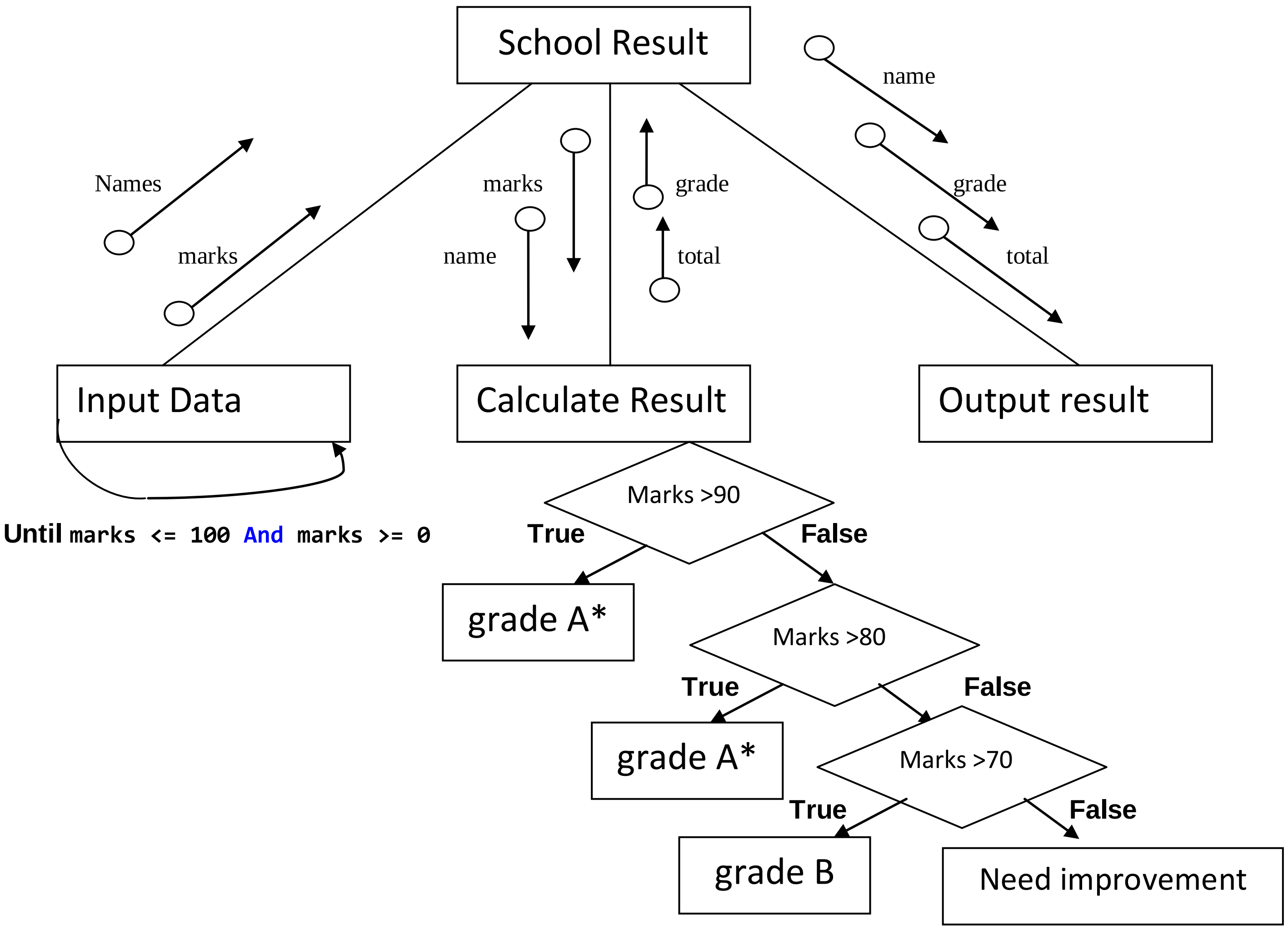
TASK 1.2

Design a modular program to implement the activity described in TASK 1.
Produce a structure chart to represent the modular structure of the solution.
The structure chart should address:

- the sequence of module execution
- any module selection or iteration
- the parameters that are passed between the modules.

ANSWER

Activity examples is taken from different areas, such as: **School or Collage**



TASK 1.3

For each module, decide whether the solution should be implemented as a procedure or a function. Justify your choices.

Produce **pseudocode** headers for each module.

ANSWER

```
BEGIN

    DECLARE name : STRING    // Global variable declared to use in all procedures
    DECLARE marks: Integer

    PROCEDURE input_sub()
        OUTPUT("Enter your Name")
        INPUT name

      REPEAT
        OUTPUT("Enter your marks")
        INPUT marks
      UNTIL marks <= 100 And marks >= 0
    END PROCEDURE


    PROCEDURE Calculation()
        If marks >= 90 Then
            grade = "Grade A*"
        ElseIf marks >= 80 Then
            grade = "Grade A"
        ElseIf marks >= 70 Then
            grade = "Grade B"
        Else
            grade = "Improvement needed"
        End IF
    END PROCEDURE

    PROCEDURE output_sub()
        OUTPUT ("The Student " , name , " scored " , marks , " has " , grade)
    END PROCEDURE


  CALL input_sub()
  CALL  Calculation()
  CALL  output_sub()

END
```

Justification of choices

- **Procedures are used in program**
- **Procedure may or may not return a value whereas function must return a value.**
- **Program doesn’t need value to be returned so it works perfectly with Procedures.**

Find below same Program in VB Code and its Output

```
Module Module1

    Dim name As String
    Dim marks As Integer
    Dim grade As String
    Sub input_sub()
        Console.Clear()
        Console.WriteLine("Enter your name")
        name = Console.ReadLine
        Do
            Console.WriteLine("Enter your marks")
            marks = Console.ReadLine
        Loop Until marks <= 100 And marks >= 0

    End Sub

    Sub Calculation()
        If marks >= 90 Then
            grade = "Grade A*"
        ElseIf marks >= 80 Then
            grade = "Grade A"
        ElseIf marks >= 70 Then
            grade = "Grade B"
        Else
            grade = "Improvement needed"
        End If
    End Sub

    Sub output_sub()
        Console.Write("The Student " & name & " scored " & marks & " has " & grade)
        Console.ReadLine()
    End Sub

    Sub Main()
        input_sub()
        Calculation()
        output_sub()
    End Sub

End Module
```

TASK 2 – Algorithms, arrays and pseudocode

Declare an array to store multiple pieces of data for your activity in TASK 1. For example, a 1D array of `STRING` could store the names of students in a class.

```
    DECLARE name [10] : STRING //Global variable declared to use in all procedures
    DECLARE marks[10] : Integer

    PROCEDURE input_sub()

      FOR count = 1 to 10
        OUTPUT("Enter your Name")
        INPUT name(count)
      NEXT count
```

Rest of program will be same way using One-Dimensional Array and For Loop

TASK 2.1

Design an algorithm to search for a specific value in the array and output the array index where the value is found.

Consider the differences between algorithms that search for a single rather than multiple instances of the value.

Document the algorithm using:

- structured English
- a program flowchart
- pseudocode.

TASK 2.2

Design an algorithm to manipulate data in the array, for example by sorting.

Document the algorithm as in TASK 2.1.

TASK 3 – Programs containing several components

A library maintains a list of books. The list is saved in a text file, where each line of the file represents one book.

TASK 3.1

Consider the information that should be included in the text file other than the title and the author.

TASK 3.2

Consider that this is a text file, which means that all information will be saved in `STRING` format.

Consider the implications of storing numeric information, such as the number of copies of each book.

Define the format of each line of the file so that each piece of information may be easily extracted.

TASK 3.3

Design a program in **pseudocode** that has a menu-driven interface and will perform the following tasks:

1. Add a new book to the text file. Include validation of the different pieces of information as appropriate.
2. Search for books written by a given author. Output the title of any books found, or a suitable message if no books by the given author are found.
3. End the program.

TASK 3.4 – Writing program code

Convert your pseudocode into **program code**.

TASK 3.5 – Testing

Consider how the program produced in TASK 3.4 may be tested.

TASK 4 – Algorithm modification

Additional information needs to be saved for each book, such as a publication date.

An additional task is needed to create a new file from the original file. The task will prompt the user to input the additional information for each book.

Consider possible validation of the additional information.

Write **program code** for the additional task.

Appendix

Built-in functions (pseudocode)

Each function returns an error if the function call is not properly formed.

`MID(ThisString : STRING, x : INTEGER, y : INTEGER) RETURNS STRING` returns a string of length `y` starting at position `x` from `ThisString`

Example: `MID("ABCDEFGH", 2, 3)` returns `"BCD"`

`LENGTH(ThisString : STRING) RETURNS INTEGER`

returns the integer value representing the length of `ThisString`

Example: `LENGTH("Happy Days")` returns `10`

`LEFT(ThisString : STRING, x : INTEGER) RETURNS STRING`

returns leftmost `x` characters from `ThisString`

Example: `LEFT("ABCDEFGH", 3)` returns `"ABC"`

`RIGHT(ThisString : STRING, x : INTEGER) RETURNS STRING`

returns rightmost `x` characters from `ThisString`

Example: `RIGHT("ABCDEFGH", 4)` returns `"EFGH"`

`INT(x : REAL) RETURNS INTEGER`
returns the integer part of `x`

Example: `INT(27.5415)` returns `27`

`NUM_TO_STRING(x : REAL) RETURNS STRING`
returns a string representation of a numeric value. Note:
This function will also work if `x` is of type `INTEGER`

Example: `NUM_TO_STRING(87.5)` returns `"87.5"`

`STRING_TO_NUM(x : STRING) RETURNS REAL`
returns a numeric representation of a string.
Note: This function will also work if `x` is of type `CHAR`

Example: `STRING_TO_NUM ("23.45")` returns `23.45`

Operators (pseudocode)

Operator	Description
`&`	Concatenates (joins) two strings. Example: `"Summer" & " " & "Pudding"` produces `"Summer Pudding"`
`AND`	Performs a logical `AND` on two Boolean values. Example: `TRUE AND FALSE` produces `FALSE`
`OR`	Performs a logical `OR` on two Boolean values. Example: `TRUE OR FALSE` produces `TRUE`

BLANK PAGE

BLANK PAGE

Permission to reproduce items where third-party owned material protected by copyright is included has been sought and cleared where possible. Every reasonable effort has been made by the publisher (UCLES) to trace copyright holders, but if any items requiring clearance have unwittingly been included, the publisher will be pleased to make amends at the earliest possible opportunity.

To avoid the issue of disclosure of answer- related information to candidates, all copyright acknowledgements are reproduced online in the Cambridge Assessment International Education Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cambridgeinternational.org after the live examination series.

Cambridge Assessment International Education is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which itself is a department of the University of Cambridge.