Rewarding Learning

General Certificate of Secondary Education
2023

Digital Technology

Unit 4

Digital Development Concepts

[GDG41]

THURSDAY 25 MAY, AFTERNOON

MARK SCHEME

General Marking Instructions

Introduction

Mark schemes are published to assist teachers and students in their preparation for examinations. Through the mark schemes teachers and students will be able to see what examiners are looking for in response to questions and exactly where the marks have been awarded. The publishing of the mark schemes may help to show that examiners are not concerned about finding out what a student does not know but rather with rewarding students for what they do know.

The Purpose of Mark Schemes

Examination papers are set and revised by teams of examiners and revisers appointed by the Council. The teams of examiners and revisers include experienced teachers who are familiar with the level and standards expected of students in schools and colleges.

The job of the examiners is to set the questions and the mark schemes; and the job of the revisers is to review the questions and mark schemes commenting on a large range of issues about which they must be satisfied before the question papers and mark schemes are finalised.

The questions and the mark schemes are developed in association with each other so that the issues of differentiation and positive achievement can be addressed right from the start. Mark schemes, therefore, are regarded as part of an integral process which begins with the setting of questions and ends with the marking of the examination.

The main purpose of the mark scheme is to provide a uniform basis for the marking process so that all the markers are following exactly the same instructions and making the same judgements in so far as this is possible. Before marking begins a standardising meeting is held where all the markers are briefed using the mark scheme and samples of the students' work in the form of scripts. Consideration is also given at this stage to any comments on the operational papers received from teachers and their organisations. During this meeting, and up to and including the end of the marking, there is provision for amendments to be made to the mark scheme. What is published represents this final form of the mark scheme.

It is important to recognise that in some cases there may well be other correct responses which are equally acceptable to those published: the mark scheme can only cover those responses which emerged in the examination. There may also be instances where certain judgements may have to be left to the experience of the examiner, for example, where there is no absolute correct response – all teachers will be familiar with making such judgements.

		AVAILABLE MARKS
1	**(a)** C American Standard Code for Information Interchange [1]	
	(b) C The maximum number of values that can be represented is 128 [1]	
	accept circled answers in parts **(a)** and **(b)**	

(c)

P	Q	R = P and Q	Q or R
0	0	0	0 [1]
0	1	0 [1]	1 [1]
1	0	0 [1]	0 [1]
1	1	1	1 [1]

[6]

(d)

Statement	**Tick (✓)**
Unicode can represent more characters than 7-bit ASCII code	✓
Unicode only ever uses 8-bits to represent characters	
Unicode can only represent the following characters: 'A'-'Z' and 'a' – 'z'	
Unicode uses 32-bit fixed length encoding	✓

[2] 10

2 **(a)** D 9 [1]

(b) A 0 [1]

(c) D 13 [1]

(d) A 4 [1] 4

3 **(a)**

Variable	**Data type**
FridaySales	Integer [1]
AverageSales	Real/Double/Float [1]

[2]

(b) **(i)** Any **two** from:
value that cannot be changed/does not change/remains the same [1]
while the program is running [1] given a name [1] used to hold data [1]
a memory location [1] [2]
(Do not accept "Never changes" or "does not change" on its own)

(ii) **One** from:
change to the constant is reflected throughout the program/value only needs changed in one place/easier to reassign a value [1]
This is a 1 mark answer so the whole phrase should reflect the value of a constant.

(c) Sample answer accept algorithm, python, c# or VB

TotalSales=0
AverageSales=0
Input FridaySales
Input SaturdaySales
Input SundaySales
TotalSales=FridaySales+SaturdaySales+SundaySales
AverageSales=TotalSales/3
output "The average sales for this week is" AverageSales

Initialisation of TotalSales or AverageSales [1]
Correct input of all three variables [2]/correct input of 1 or 2 variables [1]
Correct calculation of TotalSales [2]/correct calculation using 2 variables [1]
Correct calculation of average [1]
Output AverageSales[1] [7]

Assignment statements must be structured correctly - no marks for incorrect assignment statements.

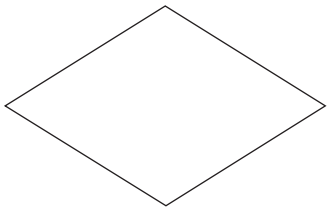
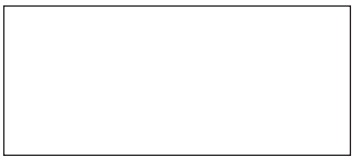
AVAILABLE MARKS

12

AVAILABLE MARKS

4 **(a)** Jack used **DECOMPOSITION [1]** to break the problem down into sub-problems. He then removed specific details which were not needed to solve the problem, this is called **ABSTRACTION [1]**.
Both of these are key elements of **COMPUTATIONAL THINKING [1]**.
Jack used **ALGORITHMS [1]** to specify the step-by-step instructions involved in the solution. [4]

(b)

Symbol	Explanation
	Any **two** from: Decision box [1] Used to ask a question [1]/used to implement if statement [1] Result of decision (is true or false/yes or no) determines the flowchart path/provides a number of paths [1] [2]
	Process box/symbol [1] Used to represent a single step/action/function/process/ sub process/calculations(s)/ describes a process taking place [1] [2]

[4]

(c) **(i)**

Statement	Tick (✓)
All syntax errors must be corrected before a program can run	✓
Syntax errors allow a program to run but the program will crash when the user enters data	
Syntax errors can only be corrected when a program is running	
An example of a syntax error is dividing by zero	
An example of a syntax error is mis-spelling a key word in the code	✓

[2]

(ii) LOGICAL ERROR[1] [1]

(iii) Two from Advantage:

- Can be designed/tested individually [1]
- Bugs can be isolated/bugs are easier to find [1]
- More efficient (program)/less storage is required [1]
- Re-useable code/can be called repeatedly [1]
- Helps code readability/Reference to improved code structure [1]

Two from Expansion:

- Shorter development time/making the coding process quicker [1]
- More robust program/allows errors to be found quickly/error free software [1]
- Code is only entered once [1]
- facilitates easier maintenance/editing [1] [4]

Two different advantages with two different expansions.

(d) Sample answer accept code or algorithm

```
do

    valid=true
    OUTPUT prompt
    INPUT option
    if option <1 OR option > 4
        valid=false
        OUTPUT error message
    end if
    clear error message
WHILE valid=false
```

or

```
Valid=false
WHILE valid!=true
    INPUT option
    if option >=1 AND option <=4
        valid=true
    ELSE
        OUTPUT error message
END WHILE
```

Accept examples using WHILE, TRUE and BREAK
Use of loop [1] Not FOR loop
Correct loop condition [1]
Input [1]
If option <1 [1] OR [1] option > 4 [1] or alternative above
Must use 'option' in condition
Loop exit logic correct [1]
Error message [1] [8]
Conditions must be correctly constructed with inequalities in the correct order

AVAILABLE MARKS

23

				AVAILABLE MARKS
5	(a)	(i)	Will ensure that more than 10 characters are entered/a minimum of 11 characters are entered [1] [1]	
		(ii)	Will ensure that data is entered/ensures that the password is not left empty [1] [1]	
		(iii)	Will ensure that the password entered contains the special symbol/ example of a special symbol at the end [1] and the capital letter [1]/ Reference to ensure that data follows a particular pattern [2]	
	(b)		Answer could contain: **Equals() [1]** Compares two strings. To check that the two passwords entered match [1] **CompareTo() [1]** (Compare two strings. To check that the two passwords entered match[1] **IndexOf() [1]** Returns the index position of first occurrence of specified character. To check that the password contains a capital letter or a special symbol [1] **Contains() [1]** Checks whether specified character or string is exists or not in the string value. To check that the password contains a capital letter or special symbol [1] **EndsWith() [1]** Checks whether specified character is the last character of string or not. To check that the password ends with a special symbol [1] **Substring() [1]** Extracts part of a string to check for capital letter or for a special symbol [1] **Length** Returns length of string. To check that the password contains more than 10 characters [1] **Count() [1]** Returns the number of times a substring appears. Used to check for capital letter or for a special symbol [1] **Trim() [1]** Removes extra whitespaces from beginning and ending of string. To ensure that there are no spaces at the beginning or end of the password entered[1] Other valid string functions with a valid explanation are acceptable. [4]	8

			AVAILABLE MARKS
6	**(a)**	OUTPUT "Enter the number of candles ordered " INPUT NumberOfCandles CostBeforeDiscount = NumberOfCandles * 3.50 [1] IF NumberOfCandles [1] > 5[1] Discount = CostBeforeDiscount/0.1 [1] * 0.1/ CostBeforeDiscount [1] Deposit = CostBeforeDiscount [1] * 0.2 TotalCostPayable = CostBeforeDiscount –Discount/Deposit [1] –Deposit/Discount [1] [8]	
	(b)	Use a file [1] to store details permanently Open when the program is not running/save to hard disk/ use program code to update/save or append file [1] [2] A sample of a file type is acceptable for [1] e.g. CSV or text.	10
7	**(a)**	Any **two** from: Binary digit [1] Can only take on the values 0 or 1 [1] Smallest unit of storage in a computer system [1] [2]	
	(b)	conversion work, accept divide by two or place value [1] 00101111/101111 [1] [2]	
	(c) **(i)**	conversion work (headings or divide by 16) [1] 4 [1] 2 [1] [3]	
	(ii)	Any **one** from: Uses less memory/storage [1] large numbers can be represented using fewer digits [1] more readable than binary patterns/can represent a larger range [1] [1]	
	(d) **(i)**	(1)10100100 correct answer (not including overflow) [1] Indicating overflow [1] Correct use of carry anywhere [1] [3]	
	(ii)	The overflow digit is (dropped/ignored/lost) [1] the result may be incorrect/not as expected [1] result exceeds 8 bits/ cannot be represented/has an extra bit/exceeds the size that can be represented [1] [2]	13

		AVAILABLE MARKS
8	**Level 0 ([0])** Answer is not worthy of credit. **Level 1 ([1]–[2])** The candidate refers to one [1], or two [2] of editing features and high level code translation and execution implying how they can help a programmer when creating and running a program. Limited examples of editing features and high level code translation are provided. The candidate makes limited use of spelling, punctuation and grammar. The meaning of the text is not always clear. The candidate displays a limited form and style appropriate to the question. The organisation of the answer is limited. **Level 2 ([3]–[4])** The candidate briefly describes, with examples, how one [3] or two [4] of editing features and high level code translation and execution can help a programmer when creating and running a program. Some clear examples of editing features and high level code translation are provided. The candidate makes satisfactory use of spelling, punctuation and grammar. The meaning of the text is usually clear. The candidate demonstrates a satisfactory form and style appropriate to the question. The organisation of the answer is satisfactory. **Level 3 ([5]–[6])** The candidate fully describes how with examples, both[5]/[6] editing features and high level code translation and execution can help a programmer when creating and running a program. Good examples of editing features and high level code translation are provided. The candidate uses a good standard of spelling, punctuation and grammar. The meaning of the text is always clear. The candidate demonstrates a good standard of form and style appropriate to the question. The organisation of the answer is good. Answers could include: Editing features – reference to: Clipboard Colour coded text/colour coded functions (note to examiners for example, syntax errors are underlined in red in the C# coding editor) Collapsible code sections Line Numbering Code Completion Tools IntelliSense/syntax error assistance Auto completion/auto indent High level code translation – reference to: The SDE provides the facility to translate the program/reference to a language specific method of running a program The program must be turned into machine code/0s and 1s/binary so that the computer can understand it/run the program [6]	6

9 **(a)** **(i)** Integer [1]

(ii) *PupilHeights [4]* [1] = 173 [1]
(Accept *PupilHeights [5]* [1]) [2]

(iii) Sample algorithm accept code also.

```
FOR I = 0 to 9 (or 1 to  10) or PupilHeights.Length
      PupilHeights [I] = 0
END FOR
```

```
Accept FOREACH e.g.
Foreach [1] pupil [1] in PupilHeights [1]
      pupil=0 [1]
```

Use of loop [1]
Correct counter [1]
Loop counter used as subscript [1]
Assignment set to 0 [1] [4]

(b)

```
maxheight=0

FOR element = 0 to 9

      IF PupilHeights [element [1]] > maxheight [1] THEN

            maxheight = PupilHeights[element] [1]
      ENDIF
END FOR
OUTPUT maxheight [1]
```

[4]

(c) **(i)**

Algorithm Statement	Order
For K = 0 to 8	2 [1]
For J = 0 To 9	1
PupilHeights[K] = PupilHeights[K + 1]	5 [1]
TEMP = PupilHeights[K]	4 [1]
IF PupilHeights[K] > PupilHeights[K + 1]	3 [1]
PupilHeights[K + 1] = TEMP	6 [1]

[5]

(ii) Insertion (sort) [1] [1]

(iii) Binary (search) [1] [1]

18

AVAILABLE MARKS

		AVAILABLE MARKS
10 **(a)**	Caitlin uses **whitebox [1]** testing to test the logic of the code. She can identify problems with the user interface, inputs and outputs by using **black box [1]** testing. **Unit [1]** testing is used to test the tiny module of code which validates the menu options presented. In order to ensure that options 3 and 4 work together Caitlin uses **integration [1]** testing. Caitlin has discovered through **system [1]** testing that the program does not meet the user requirements because she has not included the code for the Player Attendance Report. [5]	

(b)

Test data type	Explanation	Example
Valid	Used to ensure that the system operates as expected with normal data	3
Invalid	Used to ensure that the system will not accept data that is not in the range invalid/erroneous/data which does not fit the criteria [1] or other acceptable invalid example [1]	7 [1]
Extreme	Used to ensure that data at the upper and lower boundaries are accepted [1] Accept upper/lower limit Accept maximum value/minimum value in place of 'boundaries'	1/6 [1]

[4] 9

11 **(a)** **(i)** **User requirements** - Any **two** from:
Compare completed system [1]
To user requirements/needs [1]
To ensure all requirements needs are met/remind developer of requirement/to ensure they know what the system needs to do [1]
Used as a check list [1]
To ensure correct requirements are used/go through requirements [1] [2]

(ii) **Design documents** - Any **two** from:
Compare final product/show how final product differs [1]
To original design/user requirements [1]
To make sure all requirements met/to help ensure application is complete/accurate/correct [1]
To ensure the design of the application is acceptable [1]
To ensure the application will run as expected/planned [1] [2]

(b)

Statements	TRUE/FALSE
Evaluation is only carried out at the end of the development process	FALSE
Evaluation can help ensure all user requirements are met	TRUE
Evaluation can be used to help get feedback from the end user	TRUE

[3] 7

Total **120**