



**General Certificate of Secondary Education
2025**

Digital Technology

Unit 4

Digital Development Concepts

[GDG41]

TUESDAY 3 JUNE, 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.

1 (a) D The string data type can store letters and numbers. [1]

(b) C (X<Y) AND (Y>Z) [1]

(c) C IF (X*Z)<0

OUTPUT "HELLO WORLD" [1]

(d) C

A	B	C = NOT (B)	D = NOT(C AND B)
0	0	1	1
0	1	0	1
1	0	1	1
1	1	0	1

[1]

(e) B The bubble sort compares adjacent elements in an array and performs several passes through an array before the data is fully sorted. [1]

5

2 (a) 15 [1], 18 [1], 21 [1] [4]

(b)

1	2	3
2	4 [1]	6 [1]
3	6 [1]	9 [1]

[4]

(c) (i) Once/1 [1]

(ii) The loop will continue to run/infinite loop/run again [1]

(iii) Change/correct the condition/last line/reference to the condition being incorrect [1]

inches!=0/inches not equal to 0/inches<>0 [1] [2]

12

AVAILABLE MARKS

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

```

OUTPUT ("Enter the number of hours by the musician")
INPUT hoursBooked
IF hoursBooked >= 8
    Category = "Category 1"
    hourlyRate = 50.00
ELSE IF hoursBooked >= 5
    Category = "Category 2"
    hourlyRate = 60.00
ELSE IF hoursBooked >= 3
    Category = "Category 3"
    hourlyRate = 70.00
ELSE IF hoursBooked < 3 / ELSE
    Category = "Category 4"
    hourlyRate = 80.00
cost = hourlyRate * hoursBooked
OUTPUT Category
OUTPUT cost
    
```

- [1] for each correct condition to a maximum of [2]
- [1] for a correct assignment of Category
- [1] for a correct assignment of hourlyRate (accept in the cost calculation OR assignment statement) OR in output statement
- [1] for correct cost calculation hoursBooked * hourly rate
- [1] for output of Category
- [1] for output of cost

Assignment statements and conditions must be structured correctly - no marks for incorrect assignment or condition statements.
 Only 1 condition in each IF-statement

[7]

(b)

Variable	Data Type
Category	string [1]
hoursBooked	Integer/int [1]

[2]

- (c) (i) Any **two** from:
- Auto completion/auto indent [1]
 - Clipboard [1]
 - Code Completion Tools [1]
 - Collapsible code sections [1]
 - IntelliSense [1]/syntax error assistance
 - Line Numbering [1]

[2]

(ii)

Error	Syntax or logic error?
A keyword in the code has been mis-spelt.	SYNTAX [1]
When a value of 9 is entered for hoursBooked the program tells Ted that this is a Category 2 booking.	LOGIC [1]
A bracket has been omitted from a statement.	SYNTAX [1]
The cost for bookings is incorrect in some cases.	LOGIC [1]

[4]

4 (a) (i) Conversion work direct to binary or conversion to decimal then binary acceptable. This can take the form of division by two or place value. 0001 [1] 1111 [1] [2]

(ii) Conversion work [1] 31 [1]
Conversion work can show addition to achieve 31 [2]

(b) (i)

	1	1	1	1	0	1	1	0
+	1	1	1	1	1	1	1	1
Carry	1	1	1	1	1	1		
(1)	1	1	1	1	0	1	0	1

[3]

(ii) Any **two** from:
Results in overflow [1]
The overflow digit is (dropped/ignored/lost) [1]
result exceeds 8 bits/cannot be represented/has an extra bit/exceeds the size that can be represented [1] [2]

(c)

Unit of Storage	Number of bits
Byte	8 [1]
Nibble	4 [1]

[2]

(d) Any two Unicode points to a maximum of two.
Any two ASCII points to a maximum of two.

Unicode (**Max 2**)

ASCII Code (**Max 2**)

- | | |
|---|---|
| <ul style="list-style-type: none"> • represents a large number of characters (accept a number more than 65 000) [1] • supports characters from many languages/can represent emojis/ASCII cannot represent emojis [1] • can use 8/16/32 bits [1] • can occupy 1–4 bytes [1] • is a superset of ASCII [1] • requires more storage than ASCII [1] so is less efficient [1] | <ul style="list-style-type: none"> • can represent 128 characters/256 characters [1] • supports basic text/primarily English [1] • can use 7-bits/can use 8 bits [1] • can occupy 1 byte [1] • is a subset of Unicode [1] • uses less storage than unicode [1] so is more efficient [1] |
|---|---|

- 5 (a) Any **two** from:
 Each line of code is reviewed [1]
 The logic of the program/code/solution can be checked [1]
 The value of the variables can be checked/traced [1]
 Can help identify/resolve errors [1]
 Checks to ensure the expected output is produced/the program functions correctly [1]
 Helps to finalise code [1] [2]

(b)

2	6	16	-
3	6	16	-
4	6	16	6 [1]
5	6 [1]	16	-
6	8 [1]	16	-
7	8 [1]	16	-

[4]

- (c) 16 [1]

7

- 6 (a) OUTPUT "How many nights hire are required? "
 INPUT NumberOfNights
 OUTPUT "Is cooking equipment required with the tent? "
 INPUT EquipmentRequired
 IF NumberOfNights <= 7[1]
 IF EquipmentRequired = 'Y'[1]
 HireCost = 40.00[1]
 ELSE
 HireCost = 30.00[1]
 ENDIF
 TotalCost = (HireCost/NumberOfNights[1] * HireCost/NumberOfNights)
 [1] +7.00[1]
 OUTPUT TotalCost[1]
 ELSE
 OUTPUT "Error Message- Number of nights must be 7 or less"[1]
 ENDIF [9]

Alternative If statement IF EquipmentRequired='N' acceptable but assignment of HireCost must be correct.

- (b) (i) can have two values[1] True/False or Yes/No[1]/two options/ 1 or 0[1] [1]
 (ii) EquipmentRequired [1]

- (c) Any **two** from:
 Cannot be modified accidentally within the program code [1]
 Improved readability [1]
 Suitable example of built-in constant, e.g. PI [1]
 Easier maintenance [1]
 Value only needs changed in one place/change to the constant is reflected throughout the program [1]
 Type safety [1]
 Code optimisation [1] [2]

(d) (i)

Statement	TRUE/FALSE
Validation will ensure that the program never crashes.	FALSE [1]
Validation cannot tell if the data is correct.	TRUE [1]
During validation the data entered is checked to ensure it is sensible and reasonable.	TRUE [1]
Validation is carried out when data is output.	FALSE [1]

[4]

(ii) Sample answers (accept code or algorithm)

Use of loop [1] Not FOR loop
Correct loop condition [1]
Input [1]
IF EquipmentRequired !='Y'[1] AND[1] EquipmentRequired !='N'[1] or
alternative but must use 'EquipmentRequired' in condition
Loop exit logic correct [1]
Error message [1]

Sample answers (accept code or algorithm)

```
valid=false
WHILE[1]valid!=true[1]
  INPUT EquipmentRequired[1]
  OF EquipmentRequired = 'Y' [1] OR[1] EquipmentRequired = 'N'[1]
    valid=true[1]
  ELSE
    OUTPUT error message[1]
  ENDIF
END WHILE
```

Accept examples using WHILE, TRUE and BREAK (see example below)

```
do
  valid=true
  OUTPUT prompt
  INPUT EquipmentRequired
  IF EquipmentRequired !='Y' AND EquipmentRequired !='N'
    valid=false
    OUTPUT error message
  ENDIF
  clear error message
  WHILE valid=false
```

```
while [1] True: [1]
  INPUT EquipmentRequired[1]
  IF EquipmentRequired =='Y' [1]
  or[1] EquipmentRequired =='N'[1]:
    break[1]
  else:
    OUTPUT error message[1]
```

[8]

25

7 Level 0 ([0])

Answer is not worthy of credit.

Level 1 ([1]–[2])

The candidate refers to one [1], or two [2] of object-oriented or procedural programming and makes some reference to one or two features. 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 one [3] or two [4] of object-oriented or procedural programming and makes reference to at least two features. 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 both [5]/[6] object-oriented and procedural programming and makes reference to at least three features. Some comparison of the approaches is provided which shows a good understanding of each. 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:

Features – reference to:

Procedural

Use of procedures/functions or subroutines

Step-by-step instructions

Top down design

Use of functions to solve problems

Example of procedural programming language

OOP

Objects

Classes

Properties/methods

Inheritance

Encapsulation/information hiding

Both allow for some level of code re-usability

[6]

6

AVAILABLE
MARKS

8 (a)	Statement	TRUE/FALSE
	The contents of an array or list structure cannot be changed	FALSE [1]
	An array or list structure holds only one value.	FALSE [1]
	Data held in an array or list structure is stored together in memory.	TRUE [1]
	The index of an array element stores the value of that element.	FALSE [1]

[4]

(b) (i) Sample algorithm accept code also.

```
FOR i = 0 to 4 or sales.Length - 1
    sales [i] = 0
END FOR
```

Any **four** from:

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

Accept FOREACH, e.g. FOREACG [1] sale[1] in sales [1]sale=0 [1] [4]

(ii) concertOn=0

```
FOR X = 0 TO 4 [1]
    IF sales [ X ] [1] >= [1] 105
        concertOn= concertOn + 1 [1]
```

[4]

- (iii) 1. use a WHILE LOOP[1]
 2. correct condition in while loop which allows iteration through the array[1]
 3. correct use of sales as array/list in calculation, e.g. sales[x][1]
 4. correct use of loop counter as index, i.e. sale[Count][1]
 5. increment loop counter[1]
 6. correct calculation of ticketsSold[1]
 7. output ticketsSold [1] (outside of loop)
 7. output moneyCollected[1] (outside of loop)

SAMPLE ANSWER

```
WHILE [1]Count < 5/<=4 [1]
    ticketsSold=ticketsSold [1]+sales[Count][1]
    Count=Count+1 [1]
END WHILE
moneyCollected=ticketsSold*2.00[1]
OUTPUT ticketsSold[1] (must use variable name)
OUTPUT moneyCollected[1] (must use variable name) [8]
```

Accept de-dent as evidence of outside the loop
 Assignment statements must be correctly structured and in correct order

AVAILABLE
MARKS

20

9 (a)

OBSERVED OUTPUT	EXPECTED OUTPUT	FEATURES AND VARIABLES
EXTREME DATA	TEST DATA VALUES	REASON
DOCUMENT	VALID DATA	VALIDATION

A test plan identifies the **FEATURES AND VARIABLES** [1] of the software to be tested and provides a

REASON [1] for each test. **EXTREME DATA** [1] is included to test that the system can cope with data values on the boundary of a range.

VALID DATA [1] is included to test that the system operates correctly with normal

data. The test plan also specifies the **TEST DATA VALUES** [1] to be entered and the

EXPECTED OUTPUT [1] following this.

The **OBSERVED OUTPUT** [1] must also be recorded in the test plan. This will help determine if the program is working as it should. [7]

(b)

Description		
This type of testing focuses on inputs and outputs.	Black box	White box [1]
This type of testing aims to test every possible pathway through the code.	Black box	White box [1]
The tester has no knowledge of the internal code in this type of testing.	Black box	White box [1]
This type of testing is usually undertaken by a programmer.	Black box	White box [1]

[4]

11

AVAILABLE MARKS

- 10 (a)** Any **two** from:
 User requirements contains a list/checklist of what must be included [1]
 User requirements can be used to show(prove/evidence) what is missing/
 used to assess if the solution meets original plan/is full and complete [1]
 User requirements are a formal agreement between the developer and
 user [1]
 User requirements can be used to show that agreement has been
 broken [1] [2]
- (b)** Any **two** form:
 The ability of the system to handle high volumes [1]of valid /null
 invalid/exceptional data [1]
 Candidates must mention two of valid/exceptional/null [2]

Total

AVAILABLE MARKS
4
120