General Certificate of Secondary Education
2024

Digital Technology

Unit 4

Digital Development Concepts

[GDG41]

THURSDAY 6 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.

AVAILABLE MARKS

1 (a) **D** No result [1]
(b) **A** This is a square, 144 [1]
(c) **D** No result [1]
(d) **C** Shape Perimeter, 20 [1] 4

2 (a) **D** 18 19 20 21 22 23 [1]
(b) **B** 2 4 6 8 [1]
(c) **C** 12 14 16 18 [1] 3

3 (a)
1. Correct cost assignment 10 (after>50) [1]
2. Condition1 : >20/>=21 [1]
3. Correct cost assignment 15 [1]
4. Condition2 : >0/>=1/Else [1]
5. Correct cost assignment 20 [1]
6. Calculation of total cost/output of total cost [1]

Allow <=20 with cost of 20 if the solution is structured to prevent negative quantities

Assignment statements must be structured correctly
Remember that there should only be 1 condition in each IF statement
Reverse allowed but remember only one condition on each line

SAMPLE ANSWER

```
OUTPUT "Enter the group size"
INPUT groupSize
IF groupSize > 50 THEN
1.  cost = 10*groupSize/OUTPUT 10* groupSize [1]
2.              ELSE IF groupsize>=21/groupsize>20 [1] THEN
3.              cost = 15*groupSize/OUTPUT 15* groupSize [1]
4.              ELSE/ELSEIF groupsize>0/groupsize>=1 [1] THEN
5.                          cost = 20*groupSize/OUTPUT 20* groupSize [1]
6.  OUTPUT cost [1]/ credit any correct output statement in  lines 1–5 above.
```

Note that alternative solutions using one correct condition and nested if statements are acceptable if correct output is achieved [6]

(b) (i) Any **two** from:
Checking/ensuring input/data (entered) [1] against a set of criteria/rules/conditions [1]/
(ensures data) is reasonable/sensible/within specific boundaries/acceptable [1] [2]

(ii) Any **two** from:
Checks the number of characters [1]
ensure the data entered does not exceed a given length [1]
can check to ensure that null data is not entered [1]/the number of characters entered is >0 [1]/is not left blank [1] [2]

(iii) Double/Real/Float [1] [1]

AVAILABLE MARKS

(c)

```
valid = FALSE [1]
WHILE valid = FALSE [1]
          Output ("Enter the group size")
          Input groupSize
          IF groupSize>=1 [1] and [1] groupSize<=100 [1]
          valid = TRUE [1]
          Else
               Output ("Enter a value between 1 and 100")
          ENDIF
          END WHILE
```

Inequalities must be correctly structured [6] 17

4 (a)

Bit pattern	Term
0110	**NIBBLE** [1]
0	**BIT** [1]
10011001	**BYTE** [1]

[3]

(b) (i) Conversion work – accept divide by two or place value [1]
01010000 [1]/allocate [1] mark for 7 bits 1010000 [2]

(ii) Conversion work
Allocate a total of [1] for showing any one or both of the following pieces of conversion work: 0101=5 or 0000=0 [1]
50 [1] [2]

(c) (i) Conversion work [1]
170 [1] [2]

(ii) Conversion work
Allocate a total of [1] for showing any one or both of the following pieces of conversion work: 1010=A or 1010=A [1]
AA [1] [2]

(d) (i) [1] for any correctly carried value
(1) [1] 01111000 [1]
([1] for overflow and [1] for result) [3]

(ii) Any **one** from:
an additional bit is added to the binary pattern [1]
the result of the calculation is incorrect [1]
reference to exceeding maximum number that can be represented [1] [1]

(e)

A	B	C = NOT(A and B)	D = C or B
0	0	1	1
0	1	**1** [1]	1
1	0	**1** [1]	1
1	**1** [1]	**0** [1]	**1** [1]

[5] 20

5 (a) In **PROCEDURAL** [1] programming a programmer specifies **STEP BY STEP** [1] what a program must do. Instructions are carried out in a **SEQUENTIAL** [1] manner.
OBJECT ORIENTED [1] programming uses self-contained **OBJECTS** [1] which contain both programming routines or methods and the **DATA** [1] being processed. [6]

(b)

Statement	**TRUE/FALSE**
Translators can be either compilers or interpreters	**TRUE** [1]
Interpreters translate the whole program at once whilst compilers translate the program line by line	**FALSE** [1]
A compiler reports all syntax errors after attempting to compile the program	**TRUE** [1]
After a program has been compiled the machine code version of the program is stored in a separate file from the source code	**TRUE** [1]

[4] 10

6 **Level 0 [0]**
Answer is not worthy of credit.

Level 1 ([1]–[2])
The candidate correctly refers to one [1] or two [2] of logic and execution errors. 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 correctly describes one [3] or two [4] of logic and execution errors. 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 the nature of logic and execution errors [5]. Good examples of how the errors are caused together with reference to how they may be resolved is included [6]. 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 may include:
Execution errors
Occur at runtime/during execution
not having enough memory to run the program
Program will compile but crashes when executing/the program has no syntax errors
IDE will provide an error handling message/exception message or code
Occurs when an instruction includes an action that cannot be carried out
Suitable example, e.g. Divide by zero/File does not exist/array subscript out of bounds
Error handling routines can be included to prevent this type of program failure

Logic errors
Occur at runtime/during execution
Causes unexpected behaviour/causes incorrect output/the output is not correct
The program will still run/execute but will produce unexpected results/the program has no syntax errors
Problems can be detected by using the debug feature/setting breakpoints/completing a dry run/using a trace table
Suitable example, e.g. incorrect condition in an IF-Statement/Loop
When the source of the error is detected the code must be modified
Whitebox testing can detect logic errors [6] 6

7 **(a)** **(i)** An array or list structure contains data of the same data type [1]. In order to access the individual value 22 in ***votes***, the array name [1] must be used, followed by the index [1] of this element. This would be written as votes [3] [1]. [4]

(ii) Integer [1] Int [1] Do not accept numeric [1]

(b) Language independent/English like instructions which represent a solution to the problem/show solution step by step/represents the flow/logic of the solution/used to design a program [1] [1]

(c) **(i)** votes[0]=23 [1] votes[1]=25 [1] votes[2]=25 [1] votes[3]=22 [1]
votes[4]=25 [1]
[1] for each of any two correct. Maximum [2] marks.
or votes = [1] (23, 25, 25, 22, 25) [1] [2]

(ii) FOR X = 0 TO **4** [1]/len (votes) –1 [1]
IF votes[**X** [1]] = **25** [1]
allVotes= **allVotes** + 1 [1] [4]

(iii) 1. use a WHILE LOOP with correct condition [1]
2. correct use of running total for totalVotes [1]
3. correct use of votes as array/list with loop counter as index [1]
4. correct increment of loop counter [1] (inside loop)
5. output totalVotes (outside loop)

Accept
WHILE..TRUE...BREAK with a suitable condition to control the loop.
Accept WHILE in range
Accept Count++

SAMPLE ANSWER

```
1.      WHILE Count < 5/<=4 [1]
2. & 3.     totalVotes=totalVotes [1]+votes[Count] [1]
4.          Count=Count+1 [1]
        END WHILE
5.      OUTPUT totalVotes [1]
```

[5]

(iv)

```
LowestVotes = 100
X=0
Do
        IF votes[X] [1] < lowestVotes [1]   THEN
                lowestVotes = votes[X] [1]
        X = X + 1 [1]  (allow X++
While   X < 5 [1]
Output lowestVotes [1]
```

[6]

AVAILABLE MARKS

(d) (i)

Statement	**Tick (✓)**
The bubble sort compares adjacent elements and swaps them if necessary	✓[1]
The bubble sort takes each element and places it in the correct place in a sorted sub-list	
After the first pass the largest number is in the correct position in the array or list.	✓[1]
The bubble sort completes only one pass and compares adjacent elements once	
The data in the array or list will be fully sorted after n-1 passes. Where n is the number of elements in the array or list	✓[1]

[3]

(ii) PASS 1

votes

22	23	24	25	19

[1]

PASS 2

votes

23	24	25	22	19

[1]

PASS 3

votes

24	25	23	22	19

[1]

PASS 4

votes

25	24	23	22	19

[1]

(e) (i) Any **two** from:
Compares each/every item in the list [1] to a target value [1]
searches sequentially [1]
until the (target) value is found [1] [2]

(ii) Any **two** from:
Requires a sorted list [1]
Finds the mid point of the data [1]
Repeatedly reduces the search list by half [1]
If there is a match the search ends [1]
Compares midpoint to target value [1] [2]

(iii) The data is sorted [1] [1]

35

AVAILABLE MARKS

8 (a)

Data	Data type
Postcode	String [1]
Identification provided?	Boolean/Bool [1]

[2]

(b) (i) Unit [1] Tests a single module/unit of the system [1]/e.g. a function/ method [1]/small part [2]

(ii)

Test Number	Item to be tested	Reason for test	Test data	Expected outcome
1.	Name	Valid Data [1]	Any string value [1]	Value Accepted
2.	Name	Null Data [1]/ Ensure data is present [1]	Press Enter Key	Value Rejected
3.	Age	Extreme Data	14 or 18 [1]	Value Accepted
4.	Age	Invalid Data [1]	35	Value Rejected [1]/not accepted [1]
5.	Age	Valid Data [1]	Any value from 14 to 18 [1]	Value Accepted

[8]

(c)

Test type	Who should carry out the testing?	What does it test?
Black Box	Someone unfamiliar with code [1]/Programmer/ developer/user [1]	External behaviour of the code [1] The input and outputs [1] Interfaces work correctly [1] Functionality [1] (of the system) User requirements are met [1] [2]
White Box	Programmer/developer [1]	The internal logic of the program [1] The structure/conditions/pathways in the program [1] Tests each line of the code [1] [2]

[6] 18

9 (a) When evaluating a system it is important to ensure that the SOLUTION [1] meets its original DESIGN criteria [1]. This can be done by comparing it with the USER REQUIREMENTS [1].
Evaluation should occur CONTINUOUSLY [1] during the DEVELOPMENT PROCESS [1]. [5]

(b) Test using high volumes [1] of valid/invalid/exceptional [1] [2] 7

Total **120**