## UNIT 2 - COMPUTATIONAL THINKING AND PROGRAMMING

## MARK SCHEME

### Guidance for examiners

Positive marking

It should be remembered that learners are writing under examination conditions and credit should be given for what the learner writes, rather than adopting the approach of penalising him/her for any omissions. It should be possible for a very good response to achieve full marks and a very poor one to achieve zero marks. Marks should not be deducted for a less than perfect answer if it satisfies the criteria of the mark scheme.

For questions that are objective or points-based the mark scheme should be applied precisely. Marks should be awarded as indicated and no further subdivision made.

For band marked questions mark schemes are in two parts.

Part 1 is advice on the indicative content that suggests the range of computer science concepts, theory, issues and arguments which may be included in the learner's answers. These can be used to assess the quality of the learner's response.

Part 2 is an assessment grid advising bands and associated marks that should be given to responses which demonstrate the qualities needed in AO1, AO2 and AO3. Where a response is not credit worthy or not attempted it is indicated on the grid as mark band zero.

## Banded mark schemes

Banded mark schemes are divided so that each band has a relevant descriptor. The descriptor for the band provides a description of the performance level for that band. Each band contains marks.

Examiners should first read and annotate a learner's answer to pick out the evidence that is being assessed in that question. Once the annotation is complete, the mark scheme can be applied.

This is done as a two stage process.

### Stage 1 – Deciding on the band

When deciding on a band, the answer should be viewed holistically. Beginning at the lowest band, examiners should look at the learner's answer and check whether it matches the descriptor for that band. Examiners should look at the descriptor for that band and see if it matches the qualities shown in the learner's answer. If the descriptor at the lowest band is satisfied, examiners should move up to the next band and repeat this process for each band until the descriptor matches the answer.

If an answer covers different aspects of different bands within the mark scheme, a 'best fit' approach should be adopted to decide on the band and then the learner's response should be used to decide on the mark within the band. For instance if a response is mainly in band 2 but with a limited amount of band 3 content, the answer would be placed in band 2, but the mark awarded would be close to the top of band 2 as a result of the band 3 content. Examiners should not seek to mark candidates down as a result of small omissions in minor areas of an answer.

### Stage 2 – Deciding on the mark

Once the band has been decided, examiners can then assign a mark. During standardising (marking conference), detailed advice from the Principal Examiner on the qualities of each mark band will be given. Examiners will then receive examples of answers in each mark band that have been awarded a mark by the Principal Examiner. Examiners should mark the examples and compare their marks with those of the Principal Examiner.

When marking, examiners can use these examples to decide whether a learner's response is of a superior, inferior or comparable standard to the example. Examiners are reminded of the need to revisit the answer as they apply the mark scheme in order to confirm that the band and the mark allocated is appropriate to the response provided.

Indicative content is also provided for banded mark schemes. Indicative content is not exhaustive, and any other valid points must be credited. In order to reach the highest bands of the mark scheme a learner need not cover all of the points mentioned in the indicative content but must meet the requirements of the highest mark band. Where a response is not creditworthy, that is contains nothing of any significance to the mark scheme, or where no response has been provided, no marks should be awarded.

| Q | Answer | Mark | AO1 | AO2 | AO3 | Total |
|---|---|---|---|---|---|---|
| 1. | **One mark for each:** | | | | | |
| (a) | <i> | 1 | 1 | | | |
| (b) | <center> | 1 | 1 | | | |
| (c) | <hr> | 1 | 1 | | | |
| 2. | **One mark for each correct pair in the correct location:**<br>**i.e.**<br>**<h1> </h1>**<br>**<center> </center>**<br>**<b> </b>**<br>**Accept either <p> or <p> </p> (No need to close p)**<br><br>**<a href=”url”> </a> (Note http:// is required or the link will not work correctly on many devices)**<br><br>**Accept alternative tags e.g. <big></big> instead of <h1></h1>, etc**<br><br>**<ul> and <li> count as a single item as both must be used together**<br><br>**Accept alternative HTML (not CSS) solutions which work (only if the identical formatting would be achieved).** | | | | | |
| | <html><br><head><br><title> | | | | | |
| | My Fitness Tracker Homepage | 1 | | 1 | | |
| | </title> | 1 | | 1 | | |
| | </head><br><body><br><center> | | | | | |
| | | 1 | | 1 | | |
| | <h1> My Fitness Tracker | 1(center) | | 1 | | |
| | </h1> | | | | | |
| | <p> <b><i> Fitness tracking wearable technology! | 1 (h1) | | 1 | | |
| | </b></i> </p> | | | | | |
| | | 1 | | 1 | | |
| | </center> | (need<br>b i) | | | | |
| | <p>Do you need to monitor:<br><ul><br><li>your heart rate </li> | | | | | |
| | <li>your running speed </li> | 1 | | 1 | | |
| | <li>total calories burned </li> | (need | | | | |
| | </ul> | ul and li) | | | | |
| | </p> | | | | | |

| Q | Answer | Mark | AO1 | AO2 | AO3 | Total |
|---|---|---|---|---|---|---|
| | <p>All of the above in a sturdy, sporty, and comfortable design. Displays time, steps, and other stats on your wristband. Wirelessly syncs with your mobile device using Bluetooth to upload to our tracking website! </p> | | | | | |
| | <hr> | 1 (hr) | | 1 | | |
| | <p> Click the link below to find out more:<br></p> | | | | | |
| | <p> <a href="http:// www.myfitnesstracker.co.uk"><br>www.myfitnesstracker.co.uk</a></p> | 1 (a href)<br>1 (http://) | | 1<br>1 | | |
| | </body><br></html> | | | | | |
| 3(a) | **One mark for each:**<br>Local variables are declared/used (accessible) within limited parts of a program/subroutine/function/ method.<br>Whereas<br>Global variables are declared/used (accessible) throughout the entire program. | 1<br>1 | 1<br>1 | | | |
| 3(b) | **One mark for each:**<br>It is better practice to use local variables whenever a variable is not needed throughout the entire program as it uses memory only when needed<br>(OR: global variables use memory whenever the program is loaded whether the variable is needed or not)<br>It is harder to debug errors involving global variables as global variables are liable to be changed within any subroutine, even if this was not initially planned.<br>Subroutines are easier to use in other programs (improved reusability) | 1<br>1<br>1 | 1<br>1<br>1 | | | |

| Q | Answer | Mark | AO1 | AO2 | AO3 | Total |
|---|---|---|---|---|---|---|
| 4. | **One mark for each:** | | | | | 6 |
| | (a) Input a number (load into calculator/ accumulator) | 1 | | 1 | | |
| | (b) Store the value in register (mailbox) FIRST | 1 | | 1 | | |
| | (c) Input a second number (load into accumulator/calculator) replacing contents. | 1 | | 1 | | |
| | (d) Add the contents of register/mailbox FIRST to the calculator/accumulator. | 1 | | 1 | | |
| | (e) Output the value in the calculator/accumulator. | 1 | | 1 | | |
| | (f) Halt / stop execution. | 1 | | 1 | | |
| 5. | | | | | | 4 |
| | 1 Declare Subroutine LoginScreen<br>2 username is string<br>3 password is string<br>4 counter is integer<br>5 flag is boolean<br>6 set flag = false<br>7 set counter = 0<br>8 repeat<br>9 output “Type in username” | | | | | |
| | 10 **input username** | 1 | | 1 | | |
| | 11 **output “Type in password”** | 1 | | 1 | | |
| | 12 input password | | | | | |
| | 13 **if username = “User1” AND password = “Pass1” then** | 1 | | 1 | | |
| | 14 output “Username and password correct”<br>15 set flag = true<br>16 else<br>17 output “Username or password incorrect”<br>18 end if<br>19 set counter = counter + 1 | | | | | |
| | 20 **until counter = 3** | 1 | | 1 | | |
| | 21 End Subroutine | | | | | |
| 6. | | | | | | 9 |
| | **Brackets+Bold text indicate other accepted Pseudocode.**<br>**Accept i,j,k for loops; accept any other meaningful variable name.**<br>**Amendments to check for zero entered or divide by zero error (and any further validation) accepted not expected.**<br>**Line numbers not necessary. Ignore indentation or lack of it.**<br>**Accept alternative solutions as long as they provide exactly the same result.** | | | | | |

| Q | Answer | Mark | AO1 | AO2 | AO3 | Total |
|---|---|---|---|---|---|---|
| | Example<br><br>Declare totalNos=0<br>Declare currentNo=0<br>Declare maxNo =0<br>Declare minNo=999 **(or any large number)**<br>Declare total=0<br>Declare average as real=0<br><br>output “Please enter the number of integers:”<br>input totalNos<br><br>Repeat **(for i = 1 to totalNos)**<br><br>output “Enter integer:”<br>input currentNo<br><br>if currentNo >maxNo<br>then maxNo= currentNo<br>endif<br><br>if currentNo <minNo<br>then minNo= currentNo<br>endif<br><br>total=total+ currentNo<br>Until totalNos loops **(next for)**<br><br>average=total/totalNos<br><br>output “Total:”<br>output total<br>output “Average:”<br>output average<br>output “Highest:”<br>output maxNo<br>output “Lowest:”<br>output minNo<br><br>End | | | | | |
| | **<u>Marks:</u>** | | | | | |
| | Output Text | 1 | | | 1 | |
| | Input value | 1 | | | 1 | |
| | Repeat (concept of loop) | 1 | | | 1 | |
| | Selection (comparison using if) | 1 | | | 1 | |
| | Correctly adding (total=total+ currentnumber) | 1 | | | 1 | |
| | Correctly calculating average (average=total/totaNo) | 1 | | | 1 | |
| | Output a variable | 1 | | | 1 | |
| | The solution provides one correct numerical output | 1 | | | 1 | |
| | The solution provides all correct numerical outputs | 1 | | | 1 | |
| | Alternatively, a flowchart matching above award full credit. | | | | | |

<table>
<tr><th>Q</th><th>Answer</th><th>Mark</th><th>AO1</th><th>AO2</th><th>AO3</th><th>Total</th></tr>
<tr><td>7.</td><td>Overview of marking for bubble sort:</td><td></td><td></td><td></td><td></td><td>8</td></tr>
<tr><td></td><td><b>One mark</b> for applying knowledge of a swap being needed if data is out of order.</td><td></td><td></td><td>1</td><td></td><td></td></tr>
<tr><td></td><td><b>One mark</b> for applying knowledge of no swap being needed if data is in order and that the items are maintained in situ in the new list. (Accepted not expected: Bubble sort is stable.)</td><td></td><td></td><td>1</td><td></td><td></td></tr>
<tr><td></td><td><b>One mark</b> for applying knowledge of the need for additional passes through the data being required if any swaps were performed.</td><td></td><td></td><td>1</td><td></td><td></td></tr>
<tr><td></td><td><b>One mark</b> for applying knowledge that when no swaps are performed the data is in order as required and the sort has ended.</td><td></td><td></td><td>1</td><td></td><td></td></tr>
<tr><td></td><td>Worked example of bubble sort:<br>
<table><tr><td>9</td><td>5</td><td>10</td><td>2</td></tr></table>
The first two items are compared and as they are out of order they are swapped. - <b>1 mark.</b><br>
<b>Result:</b> <table><tr><td>5</td><td>9</td><td>10</td><td>2</td></tr></table>
The second and third items are compared and no swap is needed. - <b>1 mark.</b><br>
<b>Result:</b> <table><tr><td>5</td><td>9</td><td>10</td><td>2</td></tr></table>
The third and fourth items are compared and swapped. – mark already awarded above OR award mark here.<br>
<b>Result:</b> <table><tr><td>5</td><td>9</td><td>2</td><td>10</td></tr></table>
The process starts again as there were swaps performed, however there is no need to include the last item in the process as this would be in the correct position – <b>1 mark.</b><br>
<b>Result:</b> <table><tr><td>5</td><td>9</td><td>2</td><td>10</td></tr></table>
The first and second position are compared and no swap is needed. – mark already awarded above OR award mark here.<br>
<b>Result:</b> <table><tr><td>5</td><td>9</td><td>2</td><td>10</td></tr></table>
</td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

| Q | Answer | Mark | AO1 | AO2 | AO3 | Total |
|---|---|---|---|---|---|---|
| | The second and third items are compared and a swap is needed. – mark already awarded above OR award mark here. | | | | | |
| | **Result:** 5 2 9 10 | | | | | |
| | The process starts again as there were swaps performed, however there is no need to include the last two items in the process as these would be in the correct position. - mark already awarded above OR award mark here. | | | | | |
| | **Result:** 5 2 9 10 | | | | | |
| | The first and second items are compared and swapped. | | | | | |
| | **Result:** 2 5 9 10 | | | | | |
| | A final pass through would result in no swaps being made meaning the data is sorted resulting in the sort ending. – **1 mark.** | | | | | |
| | **Result:** 2 5 9 10 | | | | | |
| | Overview of marking for merge sort: | | | | | |
| | **One mark** for applying knowledge of the need to divide the list into the smallest units (1 element). (Applying the concept of divide and conquer.) | | | 1 | | |
| | **One mark** for applying knowledge of the need to compare each element with the adjacent list to sort. | | | 1 | | |
| | **One mark** for applying knowledge of the need to merge the two adjacent lists. | | | 1 | | |
| | **One mark** for applying knowledge that when the lists are merged the data is in order as required and the sort has ended. | | | 1 | | |
| | Worked example of merge sort: | | | | | |
| | 9 5 10 2 | | | | | |
| | The list is divided into the smallest elements. - **1 mark.** | | | | | |
| | **Result:** 9 5    10 2 | | | | | |
| | **Result:** 9    5    10    2 | | | | | |

| Q | Answer | Mark | AO1 | AO2 | AO3 | Total |
|---|---|---|---|---|---|---|
|  | The first and second item are compared and placed in the correct order. - **1 mark.**<br>The two adjacent elements are merged. - **1 mark.**<br>**Result:** 5 9   10   2<br>The third and fourth items are compared, placed in the correct order and merged. – mark already awarded above OR award mark here.<br>**Result:** 5 9   2 10<br>The process continues by sorting (comparing) and merging the remaining two adjacent lists. – mark already awarded above OR award mark here.<br>**Result:** 2 5 9 10<br>When the last elements are merged together the data will have been sorted and the sort has ended. – **1 mark.**<br>**Result:** 2 5 9 10 |  |  |  |  |  |

| Q | Answer | Mark | AO1 | AO2 | AO3 | Total |
|---|---|---|---|---|---|---|
| 8. | **One mark per bullet point below:** | | | | | 15 |
| (a) | World is pre-populated on load with: | | | | | |
| | • one crab | 1 | | | 1 | |
| | • two or more sharks | 1 | | | 1 | |
| | • two or more fish. | 1 | | | 1 | |
| (b) | • fish move randomly around world. | 1 | | | 1 | |
| | • sharks move randomly around world. | 1 | | | 1 | |
| | • random movement implemented using a function (such as `getrandomnumber`) | 1 | | | 1 | |
| (c) | • crab moves around world according to arrow keys. | 1 | | | 1 | |
| | • crab moves with appropriate relative speed to fish and sharks(equal to or greater than the speed of the fish and sharks) | 1 | | | 1 | |
| (d) | • fish is removed from world on collision with crab. | 1 | | | 1 | |
| (e) | • sound plays when crab and fish collide. | 1 | | | 1 | |
| (f) | • adding counter to world. | 1 | | | 1 | |
| | • counter increases when crab and fish collide. | 1 | | | 1 | |
| (g) | • counter decreases when shark and crab collide. | 1 | | | 1 | |
| | • implementation via parameter passing as opposed to wholly new method. | 1 | | | 1 | |
| (h) | • Greenfoot world saved correctly as FullAquarium | 1 | | | 1 | |
| | TOTAL | 60 | 8 | 28 | 24 | 60 |