The answers on this worksheet have been taken from the 2018 OCR GCSE Computer Science Paper 2

6-A-Day – Computer Science GCSE (p2.9-2016)

Q1

1 mark per bullet, max 6

- Input title and year from user
- Open bookcodes.txt
- Call the librarycode() function…
- … with the two parameters that match input values
- … write out **code obtained** to the text file
- Close text file

6

Example algorithm

```
title = input("enter title")
year = input ("enter year")
code = librarycode(title, year)
myFile = openWrite("bookcodes.txt")
myFile.writeLine(code)
MyFile.close()
```

Note, pseudocode shown above is an example – candidates may answer very differently, but award marks if intention can be seen.

Bullet points 3,4 and 5 could be done in one line:

```
myFile.writeLine(librarycode(title, year))
```

Do not award bullet point 3 if candidate is defining the function rather than calling it.

Allow bullet point 2 (opening text file) if correctly referred to during write operation.

Bullet point 3 must include brackets () to signify it is the function being called or indication that is being called.

Q2

1 mark per bullet, max 2.

- Function **returns** a value
- Procedure **does not return** a value

2

Allow "does not" for second mark if intention is clear (ie if it is obvious that the "not" refers to not returning a value).

Allow discussion of how returned value in a function can be used (e.g. to assign to a variable or to use this returned value in some way).

Q3

1 mark per bullet, max 4. Mark in pairs.

e.g.

- Breaks down / decomposes / modularises the problem / program // structures the program
- …making it easier to design/create/test
- …each subroutine can be tested separately

- Reuse code (in different programs)
- …quicker to develop (new) programs
- …build on existing work / use of a library of subroutines

- Avoid repetition of code (in the same program)
- …makes program shorter / smaller
- … subprogram called instead of copying/pasting.
- … quicker to develop (new) programs

- Easier to maintain
  …as code is easier to understand/read
- …as code is shorter

- Easier to debug
- …as code is shorter
- …same bugs will not have been copied to other areas of the program.

- Work can be split up in a team
- ...to suit developers' skill set
- …to work on different subprogram at the same time / develop separately

- Allows for abstraction / removes complexity
- …subprograms can be used by programmers who do not need to understand how they work.

4

Maximum of two benefits with expanstions to be marked as per question.

Allow other sensible expansions.

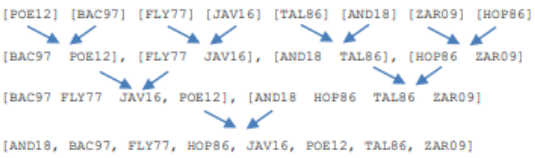
Allow expansions which cross over to other benefits (e.g. breaks down the problem / to make it easier to maintain).

Allow "can be called multiple times"

Allow "file size is smaller".

Do not allow "more efficient" without further explanation.

The answers on this worksheet have been taken from the 2018 OCR GCSE Computer Science Paper 2

Q4	1 mark per bullet, max 4. • List split into individual elements (may be done over several steps or just as a starting point) • **Merge** individual elements into **sorted** lists of size 2 • **Merge** lists of size 2 into **sorted** lists of size 4 • **Merge** lists of size 4 into final **sorted** list.	4	Candidates **can** describe how the merge sort would work rather than showing output values at each stage. Ignore intermediate steps. **Do not** give final mark for simply showing the list sorted. Must have the (correct) idea of where it being merged from previous lists. Candidates' answers describing / showing other sorting algorithms (e.g. bubble sort, insertion sort) are worth 0 marks. 
Q5	1 mark per bullet, max 2. • Faster/quicker (to sort)… • …for large lists // for lists that are **more unordered** • Has a consistent running time (for a lists of same length)… • …doesn't depend on how ordered original list is	2	Accept (correct) reference to big O notation for 2nd mark on either mark point although this is beyond scope of GCSE specification. Allow "more efficient" for BOD on first bullet point.
Q6	• 1000 0100	2	1 mark per nibble. Mark right to left.