12.3 Lists in functional programming Mark Scheme

Q1.

(a) **Mark is AO1 (understanding)**

2;
R. If more than one lozenge shaded

1

(b) **All marks AO2 (apply)**

One mark per correct row in the **Result** column:

Function call	Result
`fw [4,3]`	`12;`
`fx sales`	`[20, 50, 32];` **A.** alternative styles of bracket **R.** no brackets **R.** each element in a separate list
`fz sales`	`102;`

3

(c) **Mark is AO2 (analyse)**

Total / one day's sales value / income / revenue (for all products);
A. total / one day's profit as BOD
NE. sales, total sales

1

[5]

Q2.

(a) **Marks is for AO2 (apply)**

`10;`
A. `[10]` this time

1

(b) **Mark is for AO2 (apply)**

Function Call	Result
`map square a`	`[1,9,25]`
`filter (<10) b`	`[1,5]`
`fold (+) 0 c`	`18`

1 mark for each correct response in the **Result** column.
I. Missing brackets this time or use of incorrect type of brackets
I. If returned values are assigned to new lists eg `x = [1,9,25]`
A. [5,1] for row 2 this time

3

(c) **Mark is for AO1 (knowledge)**

A function that takes a function as an argument // returns a function as a result // takes a function as an argument and returns a function as a result;
A. "Parameter", "Input" for "Argument"
NE. A function that uses another function
R. Explanations that are specifically of the map function

1

[5]

Q3.

(a) **Marks is for AO1 (understanding)**

Head	1
Tall	[2,3,4]

1 mark for both head and tail correct.
I if brackets are missing in tail.

1

(b) **Mark is for AO2 (apply)**

[2, 4, 6, 8];

I if brackets are missing in tail.

1

(c) **All marks AO1 (understanding)**

1 mark: Explaining that map applies the function double to each list element;
1 mark: Explaining that map applies double to the head of the list;
1 mark: and then a recursive call is made on the tail of the list;

3

[5]

Examiner reports

Q1.

(a) The majority of candidates correctly identified that two of the functions made use of a higher-order functions.

(b) This question was very well tackled. Nearly 60% of candidates achieved two of the three marks and a third full marks. The most common mistake was to include brackets to indicate that a value was in a list when it was not or vice-versa.

(c) Nearly two thirds of students were able to explain that the function call calculated the total revenue generated in one day. Whilst not correct from an accounting point of view, answers that referred to the total profit were also accepted.

Q2.

(a) This question was moderately well answered with just over half of students correctly identifying that the result of applying the functions was 10. Some students gave the result as [10], which was accepted this year. Students should be aware however that, when the output of the head function should be a single value, we will not accept answers expressed as lists in the future. The most common incorrect response was 15.

(b) The purposes of the `map` and `filter` functions were fairly well understood with two thirds of students achieving two or more of the three marks, but the purpose of the `fold` function was less well known. The most common incorrect response with regard to filter was [10, 15] which suggested that students had either confused < and > or that they believed the `filter` function filtered numbers out of the result rather than filtering them into it. `fold` was poorly understood. Many students either gave the original list as the output or simply appended the 0 to the original list to generate the output. Marks were only awarded for `fold` if the result was expressed as a single value rather than a list containing a single value as the fundamental purpose of this application of the fold operation was to combine the list into one single value.

(c) The term “higher-order” function was not well understood with only just over a quarter of students correctly identifying that a higher-order function would take another function as an argument or return a function as a result. Responses that a higher-order function would use other functions were not considered mark worthy as this description could be applied to many functions. Other common mistakes were to believe that a higher-order function was simply more important than some other functions or to confuse one with a built-in function in a procedural language.