1	1	**Mark is for AO2 (apply)** **1 mark:** Both head and tail are correctly identified: • Head: `"Blackpool"` • Tail: `["Paris", "New Brighton", "Toronto"]` **I.** If quotation marks are omitted **A.** Omissions of brackets from the tail or addition of brackets to the head, this time only	**1**

1	2	**3 marks for AO2 (analysis)** The function is recursive; It splits the list up into the head and the tail; It calls itself with the tail of the list that it was called with (as an argument); Each call adds the value that is the head of the list to the total/sum of the values in the tail of the list; The recursion terminates when the list is empty (by returning 0); **Max 3**	**3**

1	3	**3 marks for AO1 (knowledge)** A function that takes a function as an argument; and/or returns a function as a result; **Max 2** **A.** “Parameter”, “Input” for “Argument” **NE.** A function that uses another function **R.** Explanations that are specifically of the `map` or `fold` functions and do not explain higher-order	**2**

1	4	**Mark is AO2 (apply)** 12;	**1**

Question			Marks
2	1	**Mark is AO2 (analyse)** `fv`; **R.** if more than one lozenge shaded	**1**

Question			Marks
2	2	**Mark is AO2 (analyse)** `fw` and `fx`; **R.** if number of shaded lozenges is not 2	**1**

Question			Marks
2	3	**All marks AO2 (apply)** **One mark** per correct row in the **Result** column:	**4**

Function call	Result
`fu 50`	10.0 **A.** value given as integer
`fv temps`	[10.0, 20.0, 35.0, 30.0] **A.** alternative styles of bracket **A.** values given as integers **R.** no brackets **R.** each element in a separate list
`fw temps`	4
`fz temps`	23.75 **A.** 95/4 **A.** average of the list the student has given on row 2 of the table (list must be more than one item) **A.** 95 divided by the answer given on row 3 of the table **A.** sum of the list the student has given on row 2 of the table (list must be more than one item) divided by the number the student has given on row 3 of the table

Question			Marks
2	4	**Mark is AO2 (analyse)** Calculates the average temperature in <u>centigrade</u> (from a list of temperatures in Fahrenheit); **NE.** calculates average of a list of numbers	**1**

Question			Marks
2	5	**Mark is AO2 (analyse)** Only one conversion is done (from Fahrenheit to centigrade) // fewer conversions (from Fahrenheit to centigrade) are performed // the function `fv` is no longer required; **A.** fewer calculations / steps / functions / function calls are required **NE.** faster execution, more efficient	**1**

Qu	Pt	Marking guidance	Total marks
3	1	**All marks AO1 (understanding)** Immutable data structures // the state of a data structure cannot be changed (after creation); Statelessness // functions do not have side-effects // all functions are pure; Functions can be distributed to servers and executed on data sets then the results can be combined // map-reduce; Higher-order functions can compose the results of processing on multiple processors/cores // functions are first-class objects; The order of execution can be determined at run-time // the order of execution can be determined by the translator/compiler/interpreter (**A.** language) // the order of execution is not defined by the program code // programs are not a sequence of instructions that must be followed in a specific order; **NE.** Suitable for parallel processing **Max 2**	**2**

Qu	Pt	Marking guidance	Total marks

4	1	**All marks are AO1 (understanding)**
		
		(Data structures/variables are immutable which means that) the state/values stored in data structures/variables cannot be changed (after they are created) // functional programming languages do not have variables;
		
		(Functions / programs are stateless which means that) functions do not have side-effects // the output of a function depends only on its inputs // functions are pure // the output of a function is not influenced by a stored state;
		
		Higher-order functions can compose the results of processing on multiple processors/cores // higher order functions can take a function as an argument and apply it to every element in a list // map-reduce can be used // functions are first-class objects and so can be passed to other functions as an argument;
		
		The order of execution can be determined at run-time // the order of execution can be determined by the translator/compiler/interpreter (**A.** language) // the order of execution is not defined by the program code // programs are not a sequence of instructions that must be followed in a specific order;
		
		Max 2
			2