**Additional answers**

1. In the first question I chose the option Int -> Int -> Int because the function “add” takes two Int parameters x y, then it makes addition of those two numbers and return Int as result.
2. (Int -> Int) -> … in type indicates that function needs to take function as input. This function has input Int and output also Int. The second part this part … -> Int tells us that returning value of function is Int.
3. If we have this list [[1,2], 2, [3,4]] and we proceed from the outside while discovering types we can say that main type is […] list of some other types inside. Those types inside are also […] lists but now lists of numbers (in this case Integers). We can also rewrite it as follows [[1,2], [2], [3,4]] so it’s more clear that type is [[Integer]]. Second value is list with one element.
4. [(["A", "B", "C"], 1000), (["C", "B"], 700)] is from outside list of elements. Inside of it there are tuples (a,b), where a is list of Strings and b is Integer. That’s why I chose [([String], Integer)]
5. ['a', '2', 'e', '~'] Is list of characters that are in Haskell represented by type Char. It’s character because of single quotation marks. Final answer is [Char].
6. I chose option “multPairs = map myMult” because other two have wrongly implemented function composition and precisely eta reduction. Normally one can also write “multPairs xs = map myMult xs” where function takes list of tuples and map function myMault across this list xs.
7. I chose option B because we need to firstly map 10% across all employees by map (\*1.1). Then we firstly calculate how much do we raise individualy by $ zipWith (\*) x y and then we zip it with original amount by zipWith (+) x. While multiplaying percentages it doesn’t matter what is first and what is second.
8. It’s a recursive function that adds value (x) y-times so basically we can say it is multiplication. The calculation is over when second parameter become 1 and the result is accumulated and returned.
9. I chose option E. Again we have recursive function that append value of second parameter and pass the rest to another recursive call until the first parameters is zero. Then it return empty array and when we get back from recursion we create list of replicated values.
10. Answer “data Tree = Leaf Int | Node Tree Tree” is correct because if we have to define Tree, we use either Leaf for single value or another Tree (recursively) so we can combine those two meta-type while building the whole Tree. This tree has always left and right branch so it is binary tree.