Question 1 [12 marks]

Consider the program

\[
\text{zw } f \ (h_1:t_1) \ (h_2:t_2) = f \ h_1 \ h_2 : \text{zw } f \ t_1 \ t_2 \\
\text{zw } _-_-_- = []
\]

(a) [4 marks] What is the inferred type of \text{zw}?

(b) [4 marks] What is the value of \text{zw} (*) [1,2,3,4] [5,4,3,2,1]

[You do not need to show your reasoning if you get the correct answer, but if you want partial marks, you need to explain your answer.]

(c) [4 marks] The standard zip function has the following behaviour:

\[
\text{zip } "abcdef" \ [1,2,3,4] \text{ evaluates to } [("a",1),("b",2),("c",3),("d",4)] \\
\text{zip } [1,2,3,4] \ [10..] \text{ evaluates to } [(1,10),(2,11),(3,12),(4,13)]
\]

Define \text{zip} using \text{zw}. You may use lambda (\_\_) and tuple constructor (, ,), but no other function. This should not be a recursive definition.

Question 2 [10 marks]

Consider the function

\[
\text{mapw } f \ p \ xs
\]

such that

\[
\text{mapw } f \ p \ [x_1, x_2, \ldots, x_n] = [f \ x_1, f \ x_2, \ldots, f \ x_k]
\]

where \( k \) is the largest index such that \( p \ x_i \) is true for all \( i \leq k \).

It should have the following behaviour:

*Main> mapw (2+) (>2) [7,6,5,4,3,2,1,21,20,19]
[9,8,7,6,5]
*Main> mapw (2+) (>2) [1,7,6,5,4,3,2,1,21,20,19]
[]
*Main> mapw (2+) (>0) [1,7,6,5,4,3,2,1,21,20,19]
[3,9,8,7,6,5,4,3,23,22,21]
*Main> mapw (^2) (\x -> x^2 < 60) [1..]
[1,4,9,16,25,36,49]

(a) [4 marks] The type of mapw is:

\[
\text{mapw :: (t -> t1) -> (t -> Bool) -> [t] -> [t1]}
\]

Explain in English (suitable for a CPSC 312 student who has just been introduced to Haskell) what this means.
(b) [6 marks] Fill in the missing values in the following recursive definition of mapw. This should use pattern matching as much as possible. You can use : and [], but no other Haskell functions.

\[
\text{mapw} \ f \ p \ __________ = __________ \\
\text{mapw} \ f \ p \ (x:xs) \\
\quad | \ __________ = __________ : ________________ \\
\quad | \ \text{otherwise} = ________________ \\
\]

Question 3 [10 marks]

In this question you can use:
list comprehensions \[
\{ f \ x \mid x <- \ \text{list}, \ \text{cond} \ x \}
\]
\[
\text{foldr} \ \oplus \ v \ [a_1, a_2, \ldots, a_n] = a_1 \oplus (a_2 \oplus (\ldots \oplus (a_n \oplus v)))
\]
\[
\text{foldl} \ \oplus \ v \ [a_1, a_2, \ldots, a_n] = (((v \oplus a_1) \oplus a_2) \oplus \ldots) \oplus a_n
\]

(a) [6 marks] Implement \(ZW\) (as defined earlier) in terms of the standard \(zip\) (described earlier) and either list comprehensions, foldr or foldl. You may use lambda (\(\_\)), (\_, \_), [ ] and : but no other built-in functions/constants. You may not use recursion.

(b) [4 marks] Given the definition:
\[
\text{delal} \ p \ \text{lst} = \text{foldr} \ (\lambda x \ y \rightarrow \text{if} \ p \ x \ \text{then} \ y \ \text{else} \ x:x:y) \ [] \ \text{lst}
\]
\[
\text{vowel} \ x = x \ '\text{elem}' \ "\text{aeiou}"
\]

What is the value of
\[
\text{delal} \ \text{vowel} \ "\text{abcef}"
\]
[You do not need to show your reasoning if you get the correct answer, but if you want partial marks, you need to explain your answer.]

Question 4 [10 marks]

(a) [3 marks] Explain the meaning of the type declaration:
\[
\mathbin{/} :: \text{Fractional} \ a \Rightarrow a \rightarrow a \rightarrow a
\]

(b) [2 marks] What is the result of entering into the Haskell prompt (in ghci):
\[
[6,7,8] : []
\]

(c) [2 marks] What is the result of entering into the Haskell prompt (in ghci):
\[
[6,7,8] : [9]
\]

(d) [3 marks] Why is lazy evaluation preferable to call-by-value? (There is no need to define either lazy evaluation or call-by-value.)