The University of British Columbia
Department of Computer Science
Midterm Examination 3 — Fall 2016

Computer Science 312
Functional and Logic Programming

Question 1 [12 marks]

Consider the program

```
zipWith _ _ [] = []
zipWith _ [] _ = []
zipWith f (h1:t1) (h2:t2) = f h1 h2 : zipWith f t1 t2
```

(a) [4 marks] What is the inferred type of `zipWith`?

(b) [4 marks] What is the value of
`zipWith (*) [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::

```
zip [1,2,3,4] [11,22,33,44]  evaluates to [(1,11),(2,22),(3,33),(4,44)]
zip [1,2,3,4] "abcdef"       evaluates to [(1,'a'),(2,'b'),(3,'c'),(4,'d')]
```

Define zip using zipWith. You may use lambda (\), and tuples, but no other function. This should not be a recursive definition.

Question 2 [10 marks]

Consider the function

```
mapWhile f p xs
```

such that

$$mapWhile\ f\ p\ [x_1, x_2, \ldots, x_n] = [f\ x_1, f\ x_2, \ldots, f\ x_k]$$

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

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

```
mapWhile :: (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 mapWhile. This should use pattern matching as much as possible. You can use : and [], but no other Haskell functions.

```
mapWhile f p _____________ = _____________

mapWhile f p (x:xs)

    | _____________    = _____________ : ___________________________________

    | otherwise = _____________
```

Question 3 [10 marks]

In this question you can use:
list comprehensions $[\, f\ x \mid x\ \texttt{<-}\ list, cond\ x\,]$
$foldr\ \oplus\ v\ [a1, a2, ..an] = a1 \oplus (a2 \oplus (... \oplus (an \oplus v)))$
$foldl\ \oplus\ v\ [a1, a2, ..an] = (((v \oplus a1) \oplus a2) \oplus ...) \oplus an$

(a) [6 marks] Implement *zipWith* (as defined earlier) in terms of *zip* (defined earlier) and either list comprehensions, foldr or foldl. You may use lambda (\) and : but no other built-in functions.

(b) [4 marks] Given the definition:

```
delal p lst = foldr (\x y -> if p x then y else x:y) [] lst
vowel x = x `elem` "aeiou"
```

What is the value of

```
delal vowel "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] Why is lazy evaluation preferable to call-by-name? (There is no need to define call-by-name.)

(b) [4 marks] Explain the meaning of the type declaration:

```
sum :: Num a => [a] -> a
```

(c) [3 marks] In Haskell every function takes a single argument. What does this mean for functions that take 2 arguments, such as (+)?