The University of British Columbia
Department of Computer Science
Midterm Examination 1 — Fall 2019

Computer Science 312
Functional and Logic Programming

Question 1 [8 marks]

Consider the program

```
dw _ [] = []
dw p (h:t)
   | p h       = dw p t
   | otherwise = (h:t)
```

(a) [4 marks] What is the inferred type of `dw`? (Use the notation that will be accepted by Haskell)

(b) [4 marks] What is the value of
`dw (/= '?') "CS312? Real fun?"`
Note that `/=` is "not equals" (defined in the Eq class).
[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 2 [10 marks]

Consider the function

```
rdint:: Integer -> String -> (Integer, String)
```

which can be used to build an integer while parsing a string. It takes the part of the integer already read, and a string, and returns the integer read from the string as well as the rest of the string. It should have the following behaviour:

```
*Mid12019> rdint 0 "123+45"
(123,"+45")
*Mid12019> rdint 1 "23+45"
(123,"+45")
*Mid12019> rdint 12 "3+45"
(123,"+45")
*Mid12019> rdint 123 "+45"
(123,"+45")
*Mid12019> rdint 0 "432"
(432,"")
*Mid12019> rdint 0 "432 and anything"
(432," and anything")
```

(a) [4 marks] Explain in English (suitable for a starting CPSC 312 student who has just been introduced to Haskell) what the type means.

(b) [6 marks] Fill in the missing values in the following recursive definition of rdint. This should use pattern matching as much as possible. You can use ", : and [], the Integer functions (+,*, etc), the functions defined below, but no other Haskell functions. Hint: the numerical value of "123" is (1*10+2)*10+3.

```
-- ch2dig converts character to number (assuming the character is a digit)
ch2dig ch = fromIntegral (fromEnum ch - fromEnum '0')
-- isdigit ch is True is the ch is a digit
isdigit ch = ch >=  '0' &&  ch <=  '9'

rdint n [] = ____________________________

rdint n (h:t)

    | isdigit h = ____________________________

    | otherwise = ____________________________
```

Question 3 [8 marks]

In this question you can use:
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) [4 marks] Filter takes a function and a list and returns the elements of the list for which the function returns true. Implement *filter* in terms of foldr or foldl. You may use if-then-else, lambda (\) and : but no other built-in functions. You cannot use recursion. For example:

```
*Mid12019> filter (>3) [4,2,7,3,1,8]
[4,7,8]
*Mid12019> filter (<3) [4,2,7,3,1,8]
[2,1]
```

(b) [4 marks] Given the definition:

```
dod p lst = foldr (\x y -> x : p x : y) [] lst
```

What is the value of

```
dod (*10) [1..5]
```

[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 [6 marks]

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

(b) [3 marks] Explain the meaning of the type declaration (suitable for a student who is just learning Haskell; explain all of the parts):

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

Question 5 [10 marks]

David wanted to build a calculator using `rdint` (defined earlier). He wants to read the first number in a string, and pass that number and the rest of string to his function `calcin`. He defined the functions:

```
calc st = calcin n r
          where (n,r) = (rdint 0 st)
calcin:: Integer -> String -> Integer
rdint:: Integer -> String -> (Integer, String)
```

(a) [4 marks] He was not happy because he thought the "where" in the definition of calc was inelegant, and thought there should be a function that takes a pair of arguments (like what is returned by `rdint`) and passes them onto a function that requires 2 arguments. Define a function `myunc` so that the following call will work the same as the above defintion. (Do not use "where" or "let" in your solution, you can only use the operator ",").

```
calc st = myunc calcin (rdint 0 st)
```

(b) [6 marks] His defintion of `calcin` is

```
calcin n [] = n

calcin n ('+':r) = calcin (n+v) t where (v,t) = rdint 0 r

calcin n ('-':r) = calcin (n-v) t where (v,t) = rdint 0 r
```

He is happy because `calc "132-22+10"` returns 120, which is correct. But is not happy because calc is not flexible enough when confronted with spaces. Show how the program can be changed to ignore spaces around operators. You can use any functions defined previously in the exam, but should only modify `calcin`. Show, on the code above, how to change `calcin` to ignore (arbitrarily many) spaces around operators. Be specific.