

---

## Final Exam — Extra Practice Questions

---

### Question 1

Are each of the following valid Haskell expressions? If so, what are their values? If not, why are they invalid?

```
length ("ka", "ki", "ku", "ke", "ko")
map prod [[5, 4, 3], 2, 1]
filter (== []) [[], [0], [], [0], []]
if 2 > 3 then ['a', 'n', 't'] else ("cater", "pillar")
elem 5 [1..10]
map isUpper
```

### Question 2

Each of the following function definitions are missing their type signatures. What are their types?

```
countDown 0          = []
countDown n          = n : countDown (n - 1)

zip [] _             = []
zip _ []             = []
zip (x:xs) (y:ys)    = (x, y) : zip xs ys

apply f x            = f x
compose f g x        = f (g x)
compose3 f g h x    = f (g (h x))

wibble f g x 0       = []
wibble f g x n       = f x : wibble g f x (n - 1)
```

### Question 3

The binary search trees discussed in lectures were represented by the following Haskell data type.

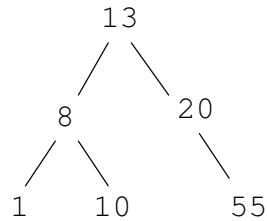
```
data Tree a
  = Null
  | Node a (Tree a) (Tree a)
```

Draw the tree represented by the following expression.

```
(Node 50 (Node 25 Null Null)
         (Node 60 (Node 55 Null Null)
                (Node 70 Null Null)))
```

#### Question 4

Using the data type from the previous question, write down the Haskell expression that represents the following tree.



#### Question 5

Describe the primary benefit of binary search trees over lists. Under what circumstances do binary trees behave poorly?

#### Question 6

The following functions have been disguised by giving them nonsensical names. Work out their type signatures, describe the behavior of each function, and suggest more meaningful names.

```
peach [] = []
peach ((x1,x2):xs) = x1 : x2 : peach xs
```

```
pear [] = []
pear (x:[]) = []
pear (x1:x2:xs) = (x1,x2) : pear xs
```

```
apple [] x = []
apple (f:fs) x = f x : apple fs x
```

```
orange ((f, x) : rs)
  | f x = x : x : orange rs
  | otherwise = x : orange rs
```

#### Question 7

The repeat function creates an (infinite) list of some value. The index function returns the  $n$ th element of a list.

```
repeat :: a -> [a]
repeat c = c : repeat c

index :: [a] -> Int -> a
index (x:xs) 0 = x
index (x:xs) n = index xs (n-1)
```

Prove the following property using induction:

$$\text{index (repeat } c) \ n = c$$

### Question 8

Here is a Java enumeration which represents the days of the week:

```
enum Day = { MON , TUE , WED , THU , FRI , SAT , SUN };
```

Write a function

```
Boolean isWeekend(Day day)
```

which checks whether the provided day is part of the weekend.

### Question 9

Using the `isWeekend` function from the previous question, write another function

```
Boolean containsWeekend(ArrayList<Day> days)
```

which accepts a list of days, and checks whether *any* of the days in that list are a part of the weekend.

### Question 10

In this course we have mentioned the word *abstraction* frequently. Give three examples of where abstraction is used in programming.

### Question 11

When defining functions we often use the concept of *recursion*. What does it mean for a function to be recursive? Why must the `length` function be recursive? Give a counter example of a function which does not use recursion.

### Question 12

Given the following two type signatures:

```
bearer  :: [a] -> Bool
brooder :: Eq a => [a] -> Bool
```

What does the `Eq` part of the type of `brooder` mean? What does this say about the definition of `brooder`? Invent definitions for both functions which have the required types.

### Question 13

My pets:

- I have a pet cat named “Floyd” who is grey in color and has four legs.
- I have a pet spider named “Fang” who is black in color and has eight legs.
- I have a pet fish named “Fredrick” who is green in color and has no legs.

Write down the data types you would use to represent this information in Haskell. Call the type which represents each pet `Pet`.

Write an expression for the list of my pets using the above data types.

Write a function

```
greenPets :: [Pet] -> [Pet]
```

which takes a list of pets and returns a list containing just the pets which are green in color.