This is a closed book exam. No calculators, cellphones, laptops, or other aids are permitted. Answer every question in the space that has been provided. You must show all your work without skipping steps; correct answers that are presented without justification may receive a mark of zero.

Student Name _____

Student Number _____

---

1. [HASKELL] Since following data type can be used to represent a list of integers…

```
data ListOfInts = EmptyList | Single Int | Cons ListOfInts Int Int
```

…how could you write a function that computes the arithmetic mean (i.e., sum of all the elements divided by the number of elements) of a list encoded using this type? The type declaration has been provided below, but you will likely want to implement your own `sum` and `length` helper functions. Please assume that the arithmetic mean of a list of length 0 is defined as 0.

```
avrage :: ListOfInts -> Float
```

This is a closed book exam. No calculators, cellphones, laptops, or other aids are permitted. Answer every question in the space that has been provided. You must show all your work without skipping steps; correct answers that are presented without justification may receive a mark of zero.

Student Name _____

Student Number _____

---

2. [HASKELL] Consider the following custom data types for storing information about movie directors (specifically an association between director name and movie title) and about release years (specifically an association between movie titles and release years).

```
directors_db :: [ ( [Char] , [Char] ) ]
directors_db = [ ("Carpenter", "Assault on Precinct 13"), ("Carpenter", "The Thing"), ("Carpenter", "Big Trouble in Little China"), ("Tarantino", "Inglourious Basterds"), ("Tarantino", "Django Unchained"), ("Tarantino", "The Hateful Eight") ]

year_db :: [ ( [Char] , Int ) ]
year_db = [ ("Escape from New York", 1981), ("The Thing", 1982), ("Big Trouble in Little China", 1986), ("Pulp Fiction", 1994), ("Django Unchained", 2012), ("The Hateful Eight", 2015) ]
```

a) Write a recursive function that, when passed the directors_db and a director name as a string, returns a list of every movie title associated with that director. You may not use any built-in functions, nor are you permitted to use the !! operator, but you may write your own helper functions, if you wish.

b) Write a recursive function that, when passed the year_db and a decade (i.e., an integer from the range 80, 90, 0, and 10), returns a list of every movie title released in that decade. You may not use any built-in functions, nor are you permitted to use the !! operator, but you may write your own helper functions, if you wish.

c) Write a recursive function that, when passed a specific year (as an integer) and the two databases - year_db and directors_db - returns the name of one director that had a movie released that year. You need not return all the directors that had movies that year - just one - but you must use the Maybe type to account for the possibility that you are passed a year that does not exist or corresponds to a movie that isn't in the directors_db, and so the return value for the function you write must be `Maybe [Char]`.

This is a closed book exam. No calculators, cellphones, laptops, or other aids are permitted. Answer every question in the space that has been provided. You must show all your work without skipping steps; correct answers that are presented without justification may receive a mark of zero.

Student Name _____

Student Number _____

2. [*Continued from the previous page*]

This is a closed book exam. No calculators, cellphones, laptops, or other aids are permitted. Answer every question in the space that has been provided. You must show all your work without skipping steps; correct answers that are presented without justification may receive a mark of zero.

Student Name _____

Student Number _____

3. [HASKELL] Prove that `(calcProduct n == 0) = anyZero n` by using structural induction. Use the following implementations for `calcProduct`, and `anyZero` and refer to the individual lines in these implementations using the labels `P1`, `P2`, `Z1`, `Z2A`, and `Z2B`. You are expected to follow the process of structural induction as it was demonstrated in class and you must show all your work (including the line applied during each step of your equational reasoning).

```
        calcProduct :: [Integer] -> Integer
P1      calcProduct [] = 1
P2      calcProduct (h:t) = h * (calcProduct t)

        anyZero :: [Integer] -> Bool
Z1      anyZero [] = False
        anyZero (h:t)
Z2A       | h == 0 = True
Z2B       | otherwise = anyZero t
```

This is a closed book exam. No calculators, cellphones, laptops, or other aids are permitted. Answer every question in the space that has been provided. You must show all your work without skipping steps; correct answers that are presented without justification may receive a mark of zero.

Student Name _____

Student Number _____

4. [HASKELL] Recreate the following function such that it is tail-call optimized. Do not forget to include a helper function (with type declaration `[a] -> [a]`) and a new type declaration for your tail-call optimized function.

```
foo :: [a] -> [a]
foo [] = []
foo (h:t) = (bar h) : (foo t)
```

5. [HASKELL] How would you write a single Haskell function to provide the sum of every even number that appears in a list of integers? You may use the built-in `even (even :: Int -> Bool)` function to complete this question but you cannot use any other functions.

6. [HASKELL] How would you write a single expression using the higher order functions `foldr`, `map`, and/or `filter` to provide the sum of every even number that appears in a list of integers? You may use the built-in `even (even :: Int -> Bool)` function to complete this question but you cannot use any other functions.

This is a closed book exam. No calculators, cellphones, laptops, or other aids are permitted. Answer every question in the space that has been provided. You must show all your work without skipping steps; correct answers that are presented without justification may receive a mark of zero.

Student Name _____

Student Number _____

7. [PROLOG] A person would be considered your "grand aunt" if and only if that person can be defined as a sibling of one of your grandparents. If you already have working predicates for `parent` and `sibling` (i.e., `parent(X,Y)` succeeds if and only if `X` is a parent of `Y` and `sibling(X,Y)` succeeds if and only if `X` and `Y` are siblings), write a predicate `grandaunt/2` such that `grandaunt(X,Y)` succeeds if and only if `Y` is the grand aunt of `x`.

8. [PROLOG] Assuming that the user continues pressing ";" as long as possible (thereby forcing Prolog to identify all possible solutions), what is the exact output provided by the following Prolog program in response to the query `a(X)`?

```
a(1).
a(2).
a(X) :- b(X), (c(X) ; d(X)).
a(X) :- e(X).
b(X) :- e(X), f(X).
b(1).
c(1).
d(1).
d(3).
e(2).
e(3).
f(3).
```

Write the Exact Output Here