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] 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. [4.0 marks]

```
foo [] = []
foo (h:t)
   | ((bar h) == True) = (h : (foo t))
   | otherwise = (foo t)
```

```
fooHelper :: [a] -> [a]
fooHelper x = fooOptimized x []

fooOptimized :: [a] -> [a] -> [a]
fooOptimized [] acc = acc
fooOptimized (h:t) acc
  | bar h = fooOptimized t (acc ++ [h])
  | otherwise = fooOptimized t acc
```

2. [HASKELL] Since following data type can be used to represent a list of Boolean values…

```
data FList = EmptyList | Single Float | Cons FList Float Float
```

…how could you write a function that computes the sum (i.e., the result when all the elements are added together) of a list encoded using this type? The type declaration for the function you will write has been provided below. You may not use any built-in functions in the creation of your solution.

```
sahm :: FList -> Float
```

[3.0 marks]

```
sahm (EmptyList) = 0
sahm (Single f) = f
sahm (Cons arg x y) = x + y + (sahm arg)
```

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 `disjoinAll n = contains True n` by using structural induction. Use the following implementations for `disjoinAll`, and `contains` and refer to the individual lines in these implementations using the labels `D1`, `D2`, `C1`, `C2A`, and `C2B`. 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).

```
        disjoinAll :: [Bool] -> Bool
D1      disjoinAll [] = False
D2      disjoinAll (h:t) = h || (disjoinAll t)

        contains :: Bool -> [Bool] -> Bool
C1      contains x [] = False
        contains x (h:t)
C2A       | h == x = True
C2B       | otherwise = contains x t
```

[5.0 marks]

```
Base Case:                                                  disjoinAll [] = contains True []
LHS                                                                                   RHS
disjoinAll []                                                          contains True []
= False "by D1"                                                         = False "by C1"

Inductive Assumption:                                        disjoinAll t = contains True t

Inductive Case:                                  disjoinAll (h:t) = contains True (h:t)
LHS                                                                                   RHS
disjoinAll (h:t)                                                   contains True (h:t)
                                      Case 1: h == True
= h || (disjoinAll t) "by D2"                                           = True "by C2A"
= True "by domination"

                                      Case 2: h == False
= h || (disjoinAll t) "by D2"                                 = contains True t "by C2B"
= (disjoinAll t) "by identity"
= contains True t "by inductive assumption"
```

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] In your second assignment, you needed to create an algebraic data type (such as the one below) for working with three-valued logic:

```
data TLogicVal = Duno | Troo | Falz deriving Show
```

Consider, as an alternative, how a `(Maybe Bool)` could be used as an alternative to a `TLogicVal` value (using `Nothing` to represent the unknown value), and rewrite the following function…

```
tConjunction :: TLogicVal -> TLogicVal -> TLogicVal
tConjunction Troo Troo = Troo
tConjunction Falz _ = Falz
tConjunction _ Falz = Falz
tConjunction _ _ = Duno
```

…such that it has type declaration

`tConjunction :: ~~(Maybe TLogicVal) -> (Maybe TLogicVal) -> (Maybe TLogicVal)~~.`

[3.0 marks]

** CORRECTIONS **

type declaration should be (Maybe Bool) -> (Maybe Bool) -> (Maybe Bool)

```
tDisjunction :: (Maybe Bool) -> (Maybe Bool) -> (Maybe Bool)

tConjunction (Just True) (Just True) = (Just True)
tConjunction (Just False)_ = (Just False)
tConjunction _ (Just False) = (Just False)
tConjunction _ _ = Nothing
```

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 _____

---

5. [HASKELL] How would you write a single Haskell function (with type declaration) to ~~provide the maximum~~ of the odd-valued elements of a list of integers? You may use the built-in `odd` function `(odd :: Int -> Bool)` to complete this question but you cannot use any other functions. That said, you can (of course) use the > operator.

[3.0 marks]

```
                    ** CORRECTIONS **
 this (and next) should be "provide the sum" of the odd-valued elements

foo :: [Int] -> Int

foo [] = 0

foo (h:t)
  | odd h = h + foo t
  | otherwise = foo t
```

6. [HASKELL] How would you write a single expression using the higher order functions `foldr`, `map`, and/or `filter` to provide the maximum of the odd-valued elements of a list of integers? You may use the built-in `odd (odd :: Int -> Bool)` function to complete this question but you cannot use any other functions. That said, you can (of course) use the greater-than operator (i.e., >).

[4.0 marks]

```
foldr (+) 0 (filter odd ARGUMENT)
```

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] 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 `u(A)`? [3.0 marks]

```
u(1).
u(A) :- v(A), x(A).
u(2).
u(A) :- y(A).
v(1).
v(A) :- y(A); z(A).
w(1).
x(1).
x(3).
y(2).
y(3).
z(3).
```

Write the Exact Output Here

```
X = 1 ;
X = 1 ;
X = 3 ;
X = 3 ;
X = 2 ;
X = 2 ;
X = 3.
```