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] -> Int`~~) and a new type declaration for your tail-call optimized function. [4.0 marks]

```
** CORRECTIONS **
type declaration should be [Int] -> Int
bar is a commutative binary operator
```

```
foo [] = 0
foo (h:t) = bar h (foo t)
```

```
fooHelper :: [Int] -> Int
fooHelper x = fooOptimized x 0

fooOptimized :: [Int] -> Int -> Int
fooOptimized [] acc = acc
fooOptimized (h:t) acc = fooOptimized t (bar h acc)
```

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

```
data BList = EmptyList | Single Bool | Cons Bool BList Bool
```

…how could you write a function that computes the length (i.e., the number of elements) 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.

```
langth :: BList -> Float
```

[3.0 marks]

```
langth (EmptyList) = 0
langth (Single _) = 1
langth (Cons _ arg _) = 2 + (langth 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 `conjoinAll n = doesNotContain False n` by using structural induction. Use the following implementations for `conjoinAll`, and `doesNotContain` and refer to the individual lines in these implementations using the labels `C1`, `C2`, `D1`, `D2A`, and `D2B`. 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).

```
        conjoinAll :: [Bool] -> Bool
C1      conjoinAll [] = True
C2      conjoinAll (h:t) = h && (conjoinAll t)

        doesNotContain :: Bool -> [Bool] -> Bool
D1      doesNotContain _ [] = True
        doesNotContain x (h:t)
D2A       | h == x = False
D2B       | otherwise = doesNotContain x t
```

[5.0 marks]

```
Base Case:                                          conjoinAll [] = doesNotContain False []
LHS                                                                                    RHS
conjoinAll []                                                    doesNotContain False []
= True "by C1"                                                           = True "by D1"

Inductive Assumption:                                 conjoinAll t = doesNotContain False t

Inductive Case:                                 conjoinAll (h:t) = doesNotContain False (h:t)
LHS                                                                                    RHS
conjoinAll (h:t)                                                 doesNotContain False (h:t)
                                      Case 1: h == True
= h && (conjoinAll t) "by C2"                                  = doesNotContain False t "by D2B"
= conjoinAll t "by identity"
= doesNotContain False t "by inductive assumption"

                                      Case 2: h == False
= h && (conjoinAll t) "by C2"                                                = False "by D2A"
= False "by domination"
```

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…

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

…such that it has type declaration

`tDisjunction :: ~~(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)

tDisjunction (Just False) (Just False) = (Just False)
tDisjunction (Just True) _ = (Just True)
tDisjunction _ (Just True) = (Just True)
tDisjunction _ _ = 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 product of the squares of the elements of a list of integers? (i.e., Square every element of the list and then multiply them all together). You cannot use any other functions to complete this question, but you can (of course) use the multiplication and exponentiation operators (i.e., * and **).

[3.0 marks]

```
** CORRECTIONS **
the exponentiation operator should be ^, not **
```

```
foo :: [Int] -> Int

foo [] = 1

foo (h:t) = (h ^ 2) * foo t
```

6. [HASKELL] How would you write a single expression using the higher order functions `foldr`, `map`, and/or `filter` to provide the product of the squares of the elements of a list of integers? You cannot use any other functions to complete this question, but you can (of course) use the multiplication and exponentiation operators (i.e., * and **).

[4.0 marks]

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

Write the Exact Output Here

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