Practice Mid-Semester Exam

STUDENT NAME

Q1 Instructions
0 Points

Australian National University

NATURAM PRIMUM COGNOSCERE RERUM

You must acknowledge the following **integrity pledge** before proceeding. Check the boxes and enter your personal details.

- [ ] I am committed to being a person of integrity.
- [ ] I pledge, as a member of the Australian National University community, to abide by and uphold the standards of academic integrity outlined in the ANU statement on honesty and plagiarism, and I am aware of the relevant legislation, and understand the consequences of me breaching those rules.
- [ ] I will not actively communicate in any way with anyone else during this exam. This includes asking questions in any online forum.

Read and check off the following instructions:

1. You will need to be able to upload code files to this browser from your Haskell workspace. You must also **make sure all code you upload compiles without errors**.
   - [ ] Make sure this browser window is open on the machine where you plan to work.
   - [ ] If you are working on the ANU Linux VDI this browser window should be open in Linux, not on your personal machine.
   - [ ] If this browser window is not open where you plan to work close this window now and log back into the exam in a browser on your work machine.
2. This examination is timed.
   - [ ] Note the remaining time at the top right of this screen. Set an alarm for yourself if you need one.
3. Permitted materials. This is an open book exam. You might in particular find the course website, the Prelude documentation, and the Data.List documentation useful.
   - [ ] You may use any documentation you wish but **all work must be your own**.

Save Answer

Q2 Programming
2.5 Points

Which statement is **false** about programming

- A compiler translates code from one form (source code) to another(target code).
- Declarative programming is only possible in a language that does not support imperative programming.
- Haskell supports declarative programming.
- Imperative programming is the oldest control flow paradigm.
- Machine and assembly code are imperative.

Save Answer

Q3 Haskell
2.5 Points

Which statement is **false** in Haskell?

- Calling the same function with the same arguments always results in the same output.
- Expressions are not evaluated until their results are needed.
- Functions can be inputs to other functions.
- Functions can return functions as outputs.
- The data stored in variables can change as the program runs.

Save Answer

Q4 Guarded Expressions
2.5 Points

Consider the following type signature and several definitions for that signature:

```haskell
myAnd:: Bool -> Bool -> Bool
```

- A.

```haskell
myAnd b c
  | b == c = b
  | otherwise = False
```

- B.

```haskell
myAnd b c
  | b = c
  | otherwise = not c
```

- C.

```haskell
myAnd b c
  | b /= c = False
  | otherwise = b
```

- D.

```haskell
myAnd b c
  | (b,c) == (True,True) = True
  | otherwise = False
```

- E.

```haskell
myAnd b c
  | b && c = True
  | otherwise = False
```

Which definition is **not** a correct implementation of the `&&` operator in Haskell?

- A
- B
- C
- D
- E

Save Answer

Q5 Style
2.5 Points

Which of the following is **not** a good way to document a Haskell function?

- A comment explaining the purpose of the function.
- A comment explaining the syntax of the function.
- A descriptive function name.
- A type declaration.
- One or more unit tests.

Save Answer

Q6 Sets
5 Points

Select which of the following statements are **true**.

Each correct answer **gains** you 1 point while each incorrect answer **loses** you 0.5 points.

- [ ] $\{1, 2, \ldots, 1000\}$ is a finite set.
- [ ] The set $A + B + C$ is equal to $A \times B \times C$.
- [ ] The set $A \to B \to C$ is equal to $A \to (B \to C)$.
- [ ] Given functions $f :: A \to B$ and $g :: B \to C$, there is a function $f \circ g :: A \to C$.
- [ ] Haskell functions are exactly the same as mathematical functions.

Save Answer

Q7 Lists
5 Points

Consider the following Haskell function:

```haskell
head xs = case xs of
  [] -> error "empty"
  (h:_) -> h
```

Select which of the following statements are **true**.

Each correct answer **gains** you 1 point while each incorrect answer **loses** you 0.5 points.

- [ ] A valid type signature for `head` is `head :: [a] -> a`.
- [ ] The type of `head "1234"` is `Char`.
- [ ] `head 'a'` gives the same result as `head "a"`.
- [ ] `head "ac" ++ head "bc"` returns `"ab"`.
- [ ] `head (True,False)` returns `True`.

Save Answer

Q8 Algebraic Data Types
8 Points

Upload a Haskell script that completes the following template (instructions included) for the functions `classOf` and `sameClass`.

Cut and paste this template into a working file `Animals.hs` for you to edit and test using ghci before uploading:

```haskell
module Animals where

-- Animals belong to a Class:
-- Mammals, Birds, Fish, Reptiles, or Amphibians.
-- DO NOT edit the type declarations below.
type Name = String

data Class
  = Mammals
  | Birds
  | Fish
  | Reptiles
  | Amphibians
  deriving (Eq, Show)

data Animal = Animal Name Class
  deriving (Eq, Show)

-- No type signatures have been provided below;
-- you must work these out for yourself.
--
-- | classOf:
-- Given an input of type Animal,
-- return the Class of that Animal.
--
-- Examples:
--
-- >>> classOf (Animal "Kangaroo" Mammals)
-- Mammals
classOf = undefined // TODO

-- | sameClass:
-- Given two inputs of type Animal,
-- return True if both belong to the same Class,
-- and False otherwise.
--
-- You may use your solution to classOf here,
-- but this is not required.
--
-- Examples:
--
-- >>> sameClass (Animal "Cat" Mammals) (Animal "Kangaroo" Mammal
-- True
--
-- >>> sameClass (Animal "Cat" Mammals) (Animal "Catfish" Fish)
-- False
sameClass = undefined // TODO
```

Please select file(s) Select file(s)

Save Answer

Q9 Control Structures
7 Points

Upload a Haskell script that completes the following template (instructions included) for the function `intDiv`.

Cut and paste this template into a working file `Division.hs` for you to edit and test using ghci before uploading:

```haskell
module Division where

-- | intDiv:
-- Given two Ints x and y,
-- if y exactly divides x with no remainder,
-- use a Maybe type to return x divided by y.
--
-- if y is zero, or if y does not exactly divide x,
-- return Nothing.
--
-- Recall the standard Prelude functions
-- div, for dividing Ints, and
-- rem, for calculating the remainder of an Int division.
--
-- Examples:
--
-- >>> intDiv 5 0
-- Nothing
--
-- >>> intDiv 5 2
-- Nothing
--
-- >>> intDiv 6 2
-- Just 3
intDiv :: Int -> Int -> Maybe Int
intDiv = undefined // TODO
```

Please select file(s) Select file(s)

Save Answer

Q10 Primitive Recursion and Recursion on Lists
8 Points

Upload a Haskell script that completes the following template (instructions included) for the function `intSum` and `countAppearances`.

Cut and paste this template into a working file `Numbers.hs` for you to edit and test using ghci before uploading:

```haskell
module Numbers where

-- | intSum:
-- Given an Int x as input,
-- return the sum of the Ints between zero and x
-- (NOT including itself).
--
-- For example, given 4 as input, return 1 + 2 + 3
--
-- Examples:
--
-- >>> intSum 4
-- 6
--
-- >>> intSum (-3)
-- -3
intSum :: Int -> Int
intSum = undefined // TODO

-- | countAppearances:
-- Given an Int and a list of Ints,
-- return the number of times that Int appears in the list.
--
-- Note that it is not useful to complete intSum
-- before you attempt this question.
--
-- Examples:
--
-- >>> countAppearances 3 [1,2,3,1,2,3]
-- 2
--
-- >>> countAppearances 2 [2,2,2,2,3]
-- 4
countAppearances :: Int -> [Int] -> Int
countAppearances = undefined // TODO
```

Please select file(s) Select file(s)

Save Answer

Q11 Recursion on Lists
7 Points

Upload a Haskell script that completes the following template (instructions included) for the function `removeVowels`.

Cut and paste this template into a working file `Vowels.hs` for you to edit and test using ghci before uploading:

```haskell
module Vowels where

-- | removeVowels:
-- Given a String as input,
-- return the original String with all vowels removed:
-- instances of 'a', 'e', 'i', 'o', and 'u'
-- (lower OR upper case) should be removed from the input.
--
-- No type signature for removeVowels has been provided;
-- you must work this out for yourself.
--
-- Examples:
--
-- >>> removeVowels "comp1100"
-- "cmp1100"
--
-- >>> removeVowels "Is this correct?"
-- "s ths crrct?"
removeVowels = undefined // TODO
```

Please select file(s) Select file(s)

Save Answer

Save All Answers   Submit & View Submission ›