Q1 Preamble
0 Points

Australian National University

COMP1100 Final Exam, Semester 2 2022

You must acknowledge the following **integrity pledge** before proceeding. Please read carefully and check all the boxes.

- ✓ I am committed to being a person of integrity.
- ✓ I pledge, as a member of the ANU community, to abide by and uphold the standards of academic integrity outlined in the ANU statement on honesty and plagiarism, I am aware of the relevant legislation, and understand the consequences of breaching those rules.
- ✓ I will not communicate in any way with anyone else during this exam. This includes asking questions in any online forum.
- ✓ I acknowledge that this exam is protected by copyright and that copying or sharing any of its content will violate that copyright.

Read and check off the following instructions:

1. This examination is timed and will be three and a half hours long (210 minutes).

- Note the remaining time at the top right of this screen. Set an alarm for yourself if you need one.

2. Permitted materials. This is an open book exam. You might in particular find the
course website, and the [Prelude documentation] useful.

   - You may use any materials you wish but **all work must be your own**.

4. Marks, exam weighting and hurdle:

   - This exam is worth 100 marks and 50% of your final grade.

   - A minimum score of 40 marks (20% of your final grade) is required to pass this course.

5. Redeemable Mid-Semester

   - This exam is redeemable against your mid-semester exam mark. If the final exam result scaled to 10% is higher than your mid-semester exam, your mid-semester mark will be replaced by this scaled mark.

6. Marking variation between gradescope and final grade.

Your marks may vary between what you see on gradescope and the final grade you receive in wattle. This can happen in your favour, for example: Your code does not compile due to a small syntax error. You receive no marks in gradescope, but partial marks in wattle.
This can also happen against your favour, for example: You write code that passes the tests in gradescope, but does not actually solve the underlying problem. You receive full marks in gradescope, but partial marks in wattle.

- ✓ I acknowledge that the mark I receive in gradescope may not reflect the mark I receive in wattle for this final exam.

Finally, here are some tips:

- ✓ Focus on the multiple choice and 5-mark programming questions first, these are worth 65% of the total exam marks.
- ✓ Attempt everything, we can give part marks for partial work, but we can't give any marks for blanks submissions or no response.

Q2 Sets and Functions 1 - Sums
1 Point

Given two arbitrary sets of integers A and B which of the following represents a possible *element* of their sum?

(2, 3)

2

(2, *left*)

{2, 3}

Q3 Sets and Functions 2 - Products
1 Point

Given two arbitrary sets of integers A and B which of the following represents a possible *element* of their product?

(2, 3)

2

(2, *left*)

{2, 3}

Q4 Sets and Functions 3 - Functions
1 Point

given:
$f(x) = x - 1$
$g(x) = x \times 2$
what is the result of:
$(f \circ g)(2)$

1

2

3

4

Q5 Operations and Data Types - Common Operations
1 Point

Which of the following Haskell functions is a valid test for an odd `Integer` value.

Solution A

```
f :: Integer -> Bool
f x = mod x 2 == 1
```

Solution B

```
f :: Integer -> Bool
f x = x `mod` 2 == 0
```

Solution C

```
f :: Integer -> Bool
f x = div x 2 == 0
```

Solution D

```
f :: Integer -> Bool
f x = div x 2 == 1
```

Select the correct solution below:

A

B

C

D

Q6 Operations and Data Types - Data Types
1 Point

What is the name of the highlighted syntax in the following data type declaration?

```
data Foo a b = MkFoo a b
```

Type Name

Data Type

Type Constructor

Data Constructor

Q7 Operations and Data Types - Data Types
1 Point

What is the name of the highlighted syntax in the following data type declaration?

```
data Foo a b = MkFoo a b
```

Type Name

Data Type

Type Constructor

Data Constructor

Q8 Recursive Data Types and Polymorphism - Lists
1 Point

Which of the following types represents a *custom* or *user-defined* list data type?

data List = Nil | Cons () List

data MyList a = Null | Node (MyList a) a (MyList a)

data [a] = [] | a : [a]

data List a = Nil | Cons a (List a)

Q9 Recursive Data Types and Polymorphism - Lists
1 Point

In the standard definition of a list which highlighted syntax of the definition is required for the type to be a *recursive* data type?
(i.e what part of the definition is the recursive part?)

data [a] = a : [a]

data [a] = a : [a]

data [a] = a : [a]

data [a] = a : [a]

Q10 Recursive Data Types and Polymorphism - Polymorphism
1 Point

Which of these is a valid polymorphic type?

data A Int String = B (Int -> String)

data A int string = B (int -> string)

data A = B (int -> string)

data A = B (Int -> String)

Q11 Lists, Local Definitions, and Code Quality - Code Quality
1 Point

Fill in the blank:
"___ *involves writing tests without looking at your code.*"

Black box testing

White box testing

Appropriate testing procedure

Best testing practice

Q12 Lists, Local Definitions, and Code Quality - Local Definitions
1 Point

When is it appropriate to use local definitions?

When we want to reuse a function more than once.

When we want to want to rename pieces of our code.

When we want to simplify our calculations, or write a helper function that is only used in our top-level function definition.

When we want to accumulate an intermediate computation.

Q13 Lists, Local Definitions, and Code Quality - Lists
1 Point

Given the following examples of an unknown function `f`:

```
f [1, 2, 3] ['a', 'b', 'c'] = [(1, 'a'), (2, 'b'), (3, 'c')]
f [(+), (*)] [(1, 2), (3, 4)] = [((+), (1, 2)), ((*), (3, 4))]
f [] [True, False, True] = []
f [] [] = []
```

What common Haskell function in the prelude behaves the same as `f` for the above inputs?

- tail
- merge
- join
- zip

Q14 Polymorphism - Functions
1 Point

Fill in the blank.
"*A function* `f :: bool -> bool` *is an example of a* ___ *function?*"

- polymorphic
- hylomorphic
- monomorphic
- catamorphic

Q15 Polymorphism - Functions 2
1 Point

Fill in the blank.
"*A function* `f :: Bool -> Bool` *is an example of a* ___ *function?*"

- polymorphic
- hylomorphic
- monomorphic
- catamorphic

Q16 Polymorphism - Functions 3
1 Point

Fill in the blank.
"*A function* `f :: Show a => a -> a` *is an example of a(n)* ___ *function?*"

- incorrect (this is not a valid Haskell type signature)
- ad-hoc polymorphic
- ad-hoc catamorphic
- parametrically polymorphic

Q17 Higher Order Functions 1
1 Point

What is the name of the syntax used in the definition of the following Haskell?

```
f :: a -> a -> (a, a)
f = \x -> \y -> (x, y)
```

- A smart constructor for pairs
- A polymorphic pair function
- A data constructor
- A lambda expression, or anonymous function

Q18 Higher Order Functions 2
1 Point

Which of the following definitions results in `g` being a higher order function?

- `g = map f xs`
- `g = map f`
- `g = map`
- none of the above.

Q19 Type Classes
1 Point

Given a datatype:

```
data Foo a = Bar a | Baz a | Qux a
```

Which of the following solutions defines a valid and complete `Ord` instance?

Solution 1.

```
instance Ord Foo where
  Bar x <= Baz y = True
  Baz x <= Quz y <= True
  Bar x <= Bar x <= True
  Baz x <= Baz x <= True
  Quz x <= Qux x <= True
  _ <= _ = False
```

Solution 2.

```
instance Ord a => Ord (Foo a) where
  (Bar x) > (Bar y) = x > y
  (Baz x) > (Baz y) = x > y
  (Qux x) > (Qux y) = x > y
  (Qux _) > (Baz _) = True
  (Qux _) > (Bar _) = True
  (Baz _) > (Bar _) = True
  _ > _ = False
```

Solution 3.

```
instance Ord a => Ord (Foo a) where
  max (Qux x) (Qux y) = max x y
  max _ (Qux x) = Qux x

  min (Bar x) (Bar y) = min x y
  min (Bar x) _ = Bar x
```

Solution 4.

```
instance Ord a => Ord (Foo a) where
  compare (Bar x) (Bar y) = compare x y
  compare (Baz x) (Baz y) = compare x y
  compare (Qux x) (Qux y) = compare x y
  compare (Bar _) (Baz _) = LT
  compare (Bar _) (Qux _) = LT
  compare (Baz _) (Qux _) = LT
  compare _ _ = GT
```

Select the correct solution below:

Solution 1

Solution 2

Solution 3

Solution 4

Q20 Binary Tree
1 Point

Assume the common definition of the binary tree data type used in this course.
Which of the following is a function to calculate tree height?
(where the height of the root node is always 0).

Solution 1.

```
f :: BTree a -> Integer
f Null = 0
f (Node l x r) = max (1 + f l) (f r)
```

Solution 2.

```
f :: BTree a -> Integer
f Null = 0
f (Node l x r)
  | f l <= f r = 1 + 0 * (f l) + 1 * (f l)
  | otherwise = 1 + 1 * (f l) + 0 * (f r)
```

Solution 3.

```
f :: BTree a -> Integer
f Null = 0
f (Node l x r) = 1 + f l + f r
```

Solution 4.

```
f :: BTree a -> Integer
f Null = 0
f (Node l x r) = 1 + max (f l) (f r)
```

Select the correct solution below:

Solution 1

Solution 2

Solution 3

Solution 4

Q21 Binary Search Tree
1 Point

Which of the following represents a function to find an element in a balanced binary search tree in $O(log(n))$ complexity, where n is the number of nodes in the tree?

Solution 1.

```
find :: Ord a => BTree a -> a -> Maybe a
find Null value = Nothing
find (Node l x r) value
  | x == value = Just x
  | otherwise = op (find l value) (find r value)
  where
    op :: Ord a => Maybe a -> Maybe a -> Maybe a
    op Nothing Nothing = Nothing
    op Nothing y = y
    op x Nothing = x
    op x y
      | x > y = x
      | otherwise = y
```

Solution 2.

```
find :: Ord a => BTree a -> a -> Maybe a
find Null value = Nothing
find (Node l x r) value
  | value == x = Just x
  | value < x = find l value
  | otherwise = find r value
```

Solution 3.

```
find :: Ord a => BTree a -> a -> Maybe a
find Null value = Nothing
find (Node l x r) value
  | x == value
     || find l value == Just value
     || find r value == Just value = Just value
  | otherwise = Nothing
```

Solution 4.

```
find :: Ord a => BTree a -> a -> Maybe a
find Null value = Nothing
find (Node l x r) value = Just x
```

Choose the correct solution below:

Solution 1

Solution 2

Solution 3

Solution 4

Q22 Rose Tree
1 Point

Which of the following definitions best describes a rose tree?

A tree where each node may have zero or more children.

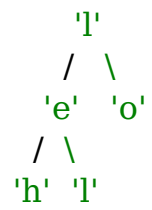
A n-ary branching tree for some n.

A recursive data structure with a Null case and a Node case.

A tree with a non-empty list of trees inside each node.

Q23 Tree Traversal
1 Point

Given the following depiction of a tree of type BTree Char:



Which of the following solutions results in the value "hello" of type String, when applied to the example above?

Solution A

```
f :: BTree a -> [a]
f Null = []
f (Node l x r) = [x] ++ f l ++ f r
```

Solution B

```
f :: BTree a -> [a]
f Null = []
f (Node l x r) = f l ++ f r ++ [x]
```

Solution C

```
f :: BTree a -> [a]
f Null = []
f (Node l x r) = f l ++ [x] ++ f r
```

Choose your answer below:

Solution A

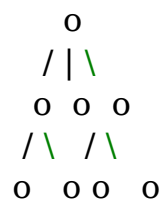
Solution B

Solution C

None of the above, these functions do not return type `String`.

Q24 Trees
1 Point

What is the most-general type of tree represented by the following diagram?



A rose tree.

A binary tree.

A game tree.

A binary search tree.

Q25 Branching Factor
1 Point

Given the definition of a Rose tree:

```
data Rose a = MkRose a [Rose a]
```

and an example tree 't':

```
t :: Rose ()
t = MkRose ()
    [MkRose ()
     [MkRose () [],
      MkRose ()
       [MkRose () [],
        MkRose () []],
        MkRose ()
         [MkRose () []]]]
```

What is the largest branching factor, of `t`?

- 4
- 3
- 2
- 1

Q26 Non-Termination
1 Point

Given an infinite list (or stream) of data:

```
double :: [Int]
double = 1 : zipWith (+) double double
```

and any predicate of type `p :: Int -> Bool`, which of the following functions terminate?

- `foldr (&&) True (map p double)`
- `foldr (&&) True (map p double) || True`
- `False && foldr (&&) True (map p double)`
- `True && foldr (&&) True (map p double)`

Q27 Complexity
1 Point

For each running time function below, select those that are *linear* ($O(n)$):

: $time\text{-}taken(n) = 4n^2 - 4n + 1$

: $time\text{-}taken(n) = 3n + 106$

: $time\text{-}taken(n) = n - 2$

: $time\text{-}taken(n) = n \log n$

Q28 Sort
1 Point

Consider the sorting algorithm which repeatedly inserts the elements of a list into an initially empty binary search tree. The insertion function used is

```
insert1 :: Ord a => a -> BTree a -> BTree a
insert1 x Null = Node Null x Null
insert1 x (Node l y r)
  | x <= y = Node (insert1 x l) y r
  | otherwise = Node l y (insert1 x r)
```

which is guaranteed to keep the elements stored in the tree's nodes in the correct order. Finally, the tree is flattened, creating a sorted list.

What is this algorithm's worst case complexity?

$O(\log n)$

$O(n)$

$O(n \log n)$

$O(n^2)$

Q29 Search
1 Point

What do we mean by a 'divide an conquer' algorithm?

- Splitting a problem in half always involves division by 2.
- Splitting a problem in half at each step can make it simpler to solve.
- Splitting a problem in half guarantees $O(logN)$ complexity.
- Splitting a problem in half creates twice the problems to solve.

Q30 Non-Termination
1 Point

Given the following Haskell code

```
data ITree a = Node (ITree a) a (ITree a)
 deriving Show

instance Functor ITree where
 fmap f (Node l x r) = Node (fmap f l) (f x) (fmap f r)

flipt :: ITree a -> ITree a
flipt (Node l x r) = Node (flipt r) x (flipt l)

flatten :: ITree a -> [a]
flatten (Node l x r) = [x] ++ flatten l ++ flatten r
```

Which of the following Haskell functions *will terminate* when given an `ITree Integer` as an argument?

- `show . (take 10) . flatten`
- `show . (fmap (+1))`
- `show . flipt`
- `show .flatten`

Q31 Laziness
1 Point

What do we mean when we say a language is a lazy language?

Haskell is only used by the laziest of programmers.

Expressions are always evaluated.

Haskell can have non-terminating terms, such as infinite lists.

Expressions are evaluated only when they are used.

Q32 Programming Questions
0 Points

There are 10 programming questions.

- 7 questions worth 5 marks (35 total)
- 3 advanced questions worth 11, 12, and 12 marks respectively (35 total)

Each question below contains a link to the appropriate Haskell file, along with some supplementary information where required.

Download the Haskell file, complete the problem specified within, and upload it according to the submission link for each problem.

The submission links are also on your gradescope dashboard with the appropriate question number and file name.

Q32.1 Greet.hs
0 Points

(Contrary to the statement above, this question is worth 5 marks!)

1. Download Greet.hs from here.
2. Complete the problem inside the Haskell file.
3. Submit your working Haskell file here.