0/23 Questions Answered

COMP1100 Final Exam

STUDENT NAME

Search students by name or email...

Q1 Acknowledgment

0 Points

Australian National University

COMP1100 Final Exam, Semester 1 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.

- [ ] 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, the Prelude documentation, the Data.List documentation, and the CodeWorld documentation useful.

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

Save Answer

Q2 True/False Questions

10 Points

Each correct answer **gains** you 2 marks, each incorrect answer **loses** you 1 mark, while a question left unanswered neither loses nor gains marks. The minimum total mark for this question is 0.

Consider the following type declaration:

```
foo :: String -> Int -> Bool
```

Q2.1
2 Points

There is a function with the same type as `foo` in the Prelude.

- [ ] True
- [ ] False

Save Answer

Q2.2
2 Points

`foo "hello" 3` could also be written as `foo ("hello", 3)`.

- [ ] True
- [ ] False

Save Answer

Q2.3
2 Points

`foo "hello" 3` could also be written as `(foo "hello") 3`.

- [ ] True
- [ ] False

Save Answer

Q2.4
2 Points

`foo ['w','o','r','l','d']` is valid Haskell.

- [ ] True
- [ ] False

Save Answer

Q2.5
2 Points

`foo . show` is valid Haskell.

- [ ] True
- [ ] False

Save Answer

Q3 Multichoice questions
20 Points

Each correct answer **gains** you 2 marks, while incorrect and unanswered questions neither gain nor lose marks.

Q3.1
2 Points

Which of the following is an **advantage** of low-level programming languages?

- [ ] Programs resemble specifications
- [ ] Their type systems help to catch errors
- [ ] They offer full control of the machine
- [ ] They offer useful abstractions for a variety of applications
- [ ] They support compositional (piece by piece) reasoning about correctness

Save Answer

Q3.2
2 Points

In the library Codeworld, which of the following is **not** a constructor for an `Event`?

- [ ] KeyRelease
- [ ] PointerClick
- [ ] PointerMovement
- [ ] TextEntry
- [ ] TimePassing

Save Answer

Q3.3
2 Points

Which of the following (mathematically identical) functions has the best **style**?

```
safeExp1 :: Int -> Int -> Maybe Int
```

```haskell
safeExp1 x y = case (x == 0 && y == 0) of
  True  -> Nothing
  False -> Just (x^y)

safeExp2 :: Int -> Int -> Maybe Int
safeExp2 x y
  | (x == 0) == False = Just (x^y)
  | (y == 0) == False = Just (x^y)
  | otherwise         = Nothing

safeExp3 :: Int -> Int -> Maybe Int
safeExp3 x y
  | x == 0 && y == 0 = Nothing
  | otherwise        = Just (x^y)

safeExp4 :: Int -> Int -> Maybe Int
safeExp4 x y
  | x == 0 && y == 0 = Nothing
  | x /= 0 || y /= 0 = Just (x^y)

safeExp5 :: Int -> Int -> Maybe Int
safeExp5 x y
  | (x == 0 && y == 0) == True = Nothing
  | otherwise                  = Just (x^y)
```

- [ ] `safeExp1`
- [ ] `safeExp2`
- [ ] `safeExp3`
- [ ] `safeExp4`
- [ ] `safeExp5`

Save Answer

Q3.4

2 Points

Which of the following is **not** mathematically identical to the identity function on lists of type `[Int]`?

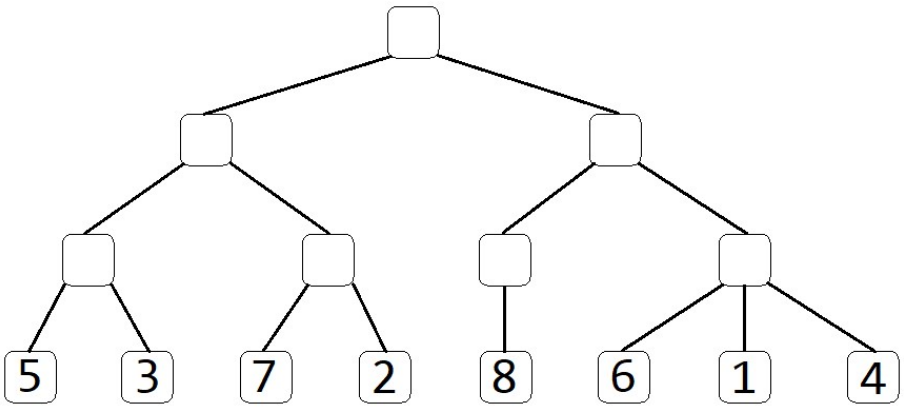
- [ ] `foldl (\x y -> x ++ [y]) []`
- [ ] `foldr (:) []`
- [ ] `filter (const True)`
- [ ] `map (+0)`
- [ ] `zipWith (+) [0..]`

Save Answer

Q3.5

2 Points

Consider the game tree below, where the leaves have been given values according to some heuristic, and each step down the tree represents a change of turn in a two player game. Assuming it is your turn, and you are trying to maximise the heuristic, what value would minimax give the root of the tree?



- [ ] 8
- [ ] 7
- [ ] 6
- [ ] 5
- [ ] 4

Save Answer

Q3.6
2 Points

Which of the below types with context are the most general (i.e. the type variables could be instantiated in the largest number of different ways)?

- [ ] Eq a => a -> a
- [ ] Eq a => a -> b
- [ ] (Eq a, Ord a) => a -> b
- [ ] (Eq a, Ord a) => a -> a
- [ ] Ord a => a -> a
- [ ] Ord a => a -> b

Save Answer

Q3.7
2 Points

Consider the Haskell function:

```
allUnique :: Eq a => [a] -> Bool
allUnique list = case list of
  []     -> True
  x:xs   -> not (elem x xs) && allUnique xs
```

What is the **best case** scenario of `allUnique` with respect to 'big O' time complexity?

- [ ] The first two elements of the list are the same
- [ ] The last two elements of the list are the same
- [ ] The input list contains all unique values and is sorted from largest to smallest
- [ ] The input list contains all unique values and is sorted from smallest to largest
- [ ] The input list is empty

Save Answer

Q3.8
2 Points

What is the **best case** time complexity of `allUnique`?

- [ ] $O(1)$
- [ ] $O(\log n)$
- [ ] $O(n)$
- [ ] $O(n \log n)$
- [ ] $O(n^2)$
- [ ] $O(n^2 \log n)$
- [ ] $O(n^3)$
- [ ] $O(n^3 \log n)$
- [ ] $O(n^4)$
- [ ] $O(2^n)$

Save Answer

Q3.9
2 Points

What is the **worst case** time complexity of `allUnique`?

- [ ] $O(1)$
- [ ] $O(\log n)$
- [ ] $O(n)$
- [ ] $O(n \log n)$
- [ ] $O(n^2)$
- [ ] $O(n^2 \log n)$
- [ ] $O(n^3)$
- [ ] $O(n^3 \log n)$
- [ ] $O(n^4)$
- [ ] $O(2^n)$

Save Answer

Q3.10
2 Points

Consider the Haskell function (recalling that `sort` is implemented in Data.List as Merge Sort):

```
ordAllUnique :: Ord a => [a] -> Bool
ordAllUnique = sortedAllUnique . sort
  where
    sortedAllUnique list = case list of
      []     -> True
```

```
        [_]    -> True
        x:y:ys -> x /= y && sortedAllUnique (y:ys)
```

What is the **worst case** time complexity of `ordAllUnique`?

- [ ] $O(1)$
- [ ] $O(\log n)$
- [ ] $O(n)$
- [ ] $O(n \log n)$
- [ ] $O(n^2)$
- [ ] $O(n^2 \log n)$
- [ ] $O(n^3)$
- [ ] $O(n^3 \log n)$
- [ ] $O(n^4)$
- [ ] $O(2^n)$

Save Answer

Q4 Programming Questions

70 Points

There are **seven** programming questions that you need to complete and submit.
You can find links to submit your programming questions on your dashboard.

Please submit by uploading **each** Haskell file to **each** question.

Each function will be marked individually for correctness and code quality. Each function is worth **5 marks**.

Please download the template Haskell files here.

Q4.1 RadioStations.hs
15 Points

Submit `RadioStations.hs` here

Save Answer

Q4.2 RepeatedPowers.hs
5 Points

Submit `RepeatedPowers.hs` here

Save Answer

Q4.3 ListFunctions.hs
10 Points

Submit `ListFunctions.hs` here

Save Answer

Q4.4 BoolFunctions.hs
10 Points

Submit `BoolFunctions.hs` here

Save Answer

Q4.5 Dogs.hs
10 Points

Submit `Dogs.hs` here

Save Answer

Q4.6 RoseTrees.hs
10 Points

Submit `RoseTrees.hs` here

Save Answer

Q4.7 Intervals.hs

10 Points

Submit `Intervals.hs` here

Save Answer

Save All Answers

Submit & View Submission ›