0/17 Questions Answered

Final Exam: Multiple-Choice and T/F Questions

STUDENT NAME

Search students by name or email

Q1 Acknowledgment

0 Points

Australian National University

NATURAM PRIMUM COGNOSCERE RERUM

COMP1100 Final Exam, Semester 1 2021

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 Haskell 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 Multi-choice

12 Points

Select one correct answer for each question. Correct answers receive full points. Incorrect answers receive no points.

Q2.1 Code Quality

2 Points

Which of the following are True?

○ One can show the absence of bugs through testing.

○ You cannot write black box tests for your own code.

○ Comments, type declarations, and unit tests should explain how functions work.

○ Randomised testing can exhaustively test special cases.

- ○ White box testing enables testing the boundaries where the program makes choices of input.

Save Answer

Q2.2 Parametric polymorphism
2 Points

Which of the following is True in Haskell?

- ○ It is always better to make functions to be polymorphic.
- ○ A polymorphic list can contain elements with different types.
- ○ The type variables should be instantiated to the same type in `id :: a -> a`
- ○ In parametric polymorphism, the function definition needs to be overridden based on the instantiation.
- ○ Any function can be written as a polymorphic function.

Save Answer

Q2.3 Lists and Recursion
2 Points

Which of the following are True in Haskell?

- ○ All algebraic data types can be recursively defined.
- ○ Recursions that compute from the right and the left of lists produce the same results.
- ○ Recursion can be achieved via multiple functions that call each other.
- ○ A pair cannot contain elements of different types.
- ○ `head lst` and `lst!!1` returns the same output.

Save Answer

Q2.4 Higher Order Functions
2 Points

Consider a function `foo` with the following type signature:

```
foo :: Double -> String -> Int
```

Suppose that `foo` can in general run without producing errors. What would happen if you tried to run `foo 0` in ghci?

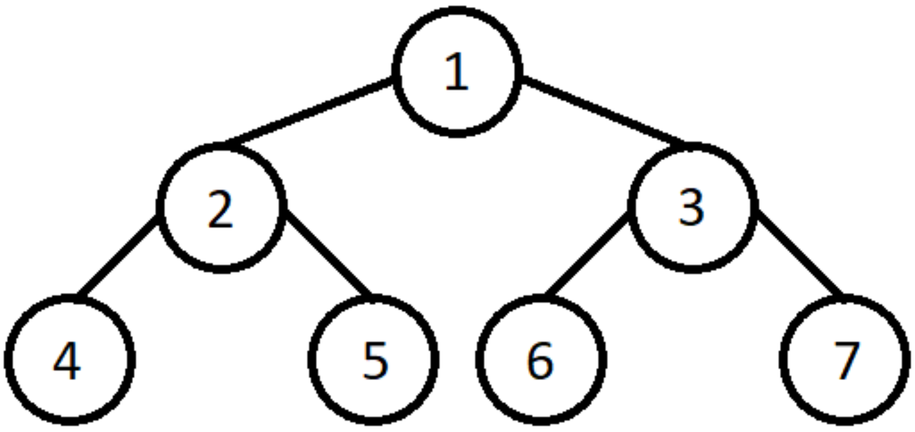
- ○ You would get an error because `0` is not a `Double`; you should have written `0.0`.
- ○ You would get an error because `foo` requires two inputs, and only one has been provided.
- ○ You would get an error because `foo` takes a function of type `Double -> String` as input.
- ○ You would get an error because function types are not instances of the `Show` typeclass.
- ○ You would not get an error.

Save Answer

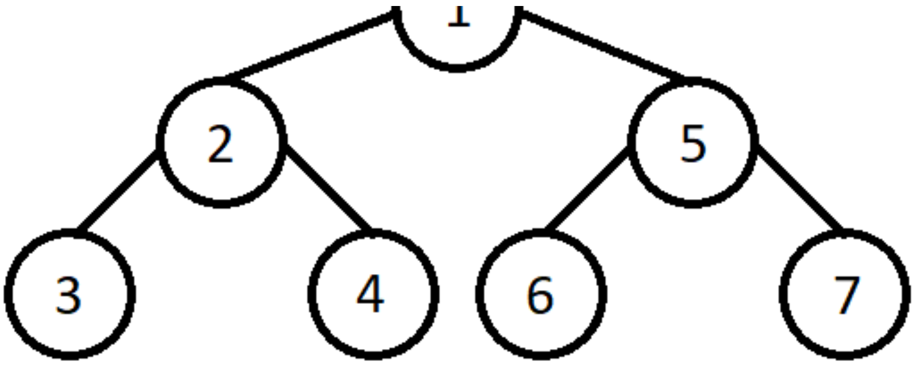
Q2.5 Binary Search Trees
2 Points

Which of the following is a binary search tree?

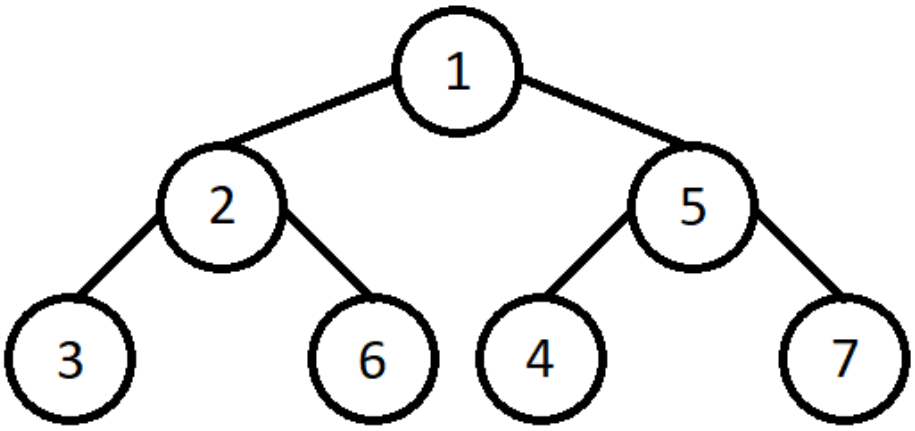
○



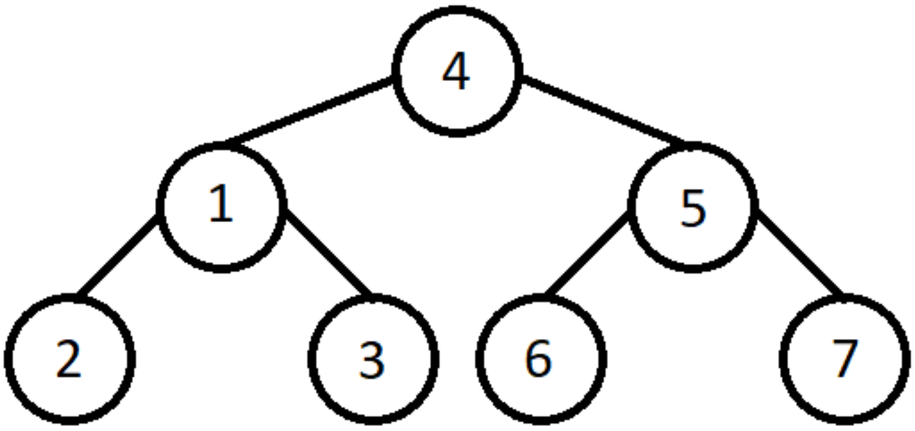
○



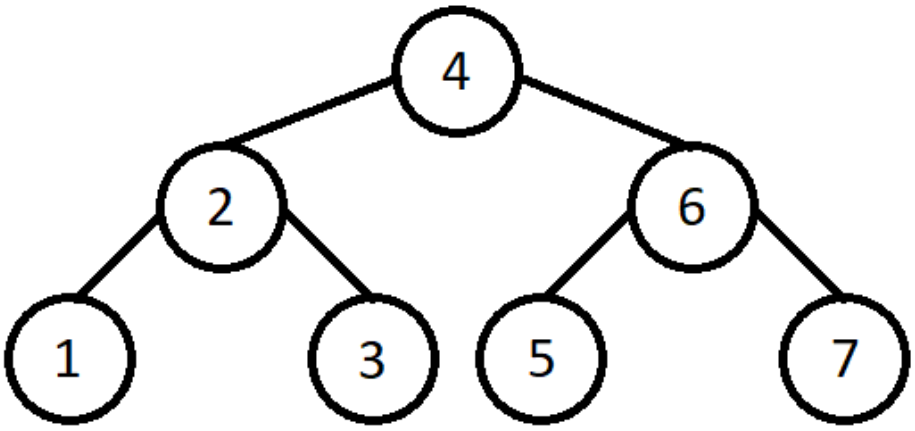
○



○



○

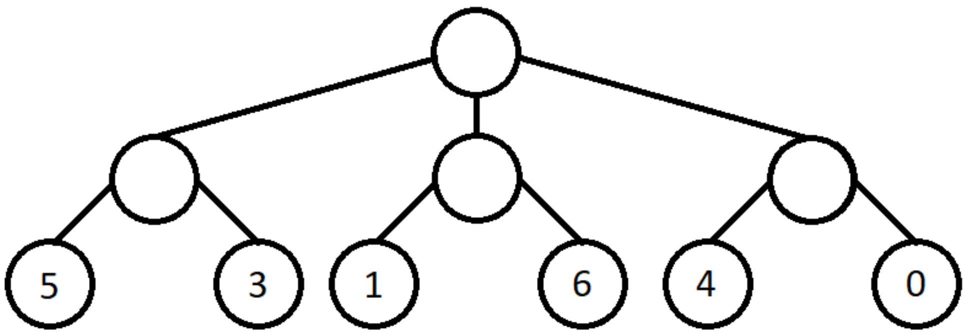


Save Answer

Q2.6 Alpha-Beta Pruning

2 Points

Suppose that you are playing a two player game with alternating turns, and are using the alpha-beta pruning algorithm with lookahead 2, and it is your turn. Suppose the complete game tree looks as follows, with the values your heuristic would calculate indicated on the bottom level:



Which of the leaves of the tree, if any, will your algorithm prune (not visit)?

- ◯ The leaves with values 1, 6, 4, and 0.
- ◯ The leaves with values 1 and 0.
- ◯ The leaf with value 6.
- ◯ The leaf with value 0.
- ◯ It will not prune any leaves.

Save Answer

Q3 True/False

10 Points

Consider the following Haskell functions defined in the `Prelude`.

```
zip :: [a] -> [b] -> [(a,b)]
unzip :: [(a,b)] -> ([a], [b])
```

Consider each of the following statements and mark them True or False.

Correct answers receive 2 points. Incorrect answers lose 1 point. No answer receives 0 points. Minimum score for this question is 0 points.

Q3.1
2 Points

We can apply the `zip` function to any two lists.

- ☐ True
- ☐ False

Save Answer

Q3.2
2 Points

Once the `zip` function is applied, we can recover the original input lists by applying the `unzip` function.

- ☐ True
- ☐ False

Save Answer

Q3.3
2 Points

`unzip . zip` is a valid function composition.

- ☐ True
- ☐ False

Save Answer

Q3.4
2 Points

We can instantiate both `a` and `b` to `Integer`.

☐ True

☐ False

Save Answer

Q3.5
2 Points

The `unzip` function always returns a pair of lists.

☐ True

☐ False

Save Answer

Q4 Complexity
8 Points

Consider each of the following functions and select its time complexity. Each function has the same best, worst, and average case, time complexity.

Select one correct answer for each question. Correct answers receive full points. Incorrect answers receive no points.

Q4.1
2 Points

What is the time complexity of the Prelude function `map`?

- $O(1)$
- $O(n)$
- $O(n^2)$
- $O(n^3)$
- $O(2^n)$

Save Answer

Q4.2

2 Points

Consider the following function:

```
-- | prefixes
--
-- Returns the list of all possible initial
-- segments of the input list.

prefixes :: [a] -> [[a]]
prefixes list = [] : case list of
  []   -> []
  x:xs -> map (x:) (prefixes xs)
```

What is the time complexity of `prefixes`?

- $O(1)$
- $O(n)$
- $O(n^2)$
- $O(n^3)$
- $O(2^n)$

Save Answer

Q4.3

2 Points

Consider the following function:

```
-- | suffixes
--
-- Returns the list of all possible final segments of the input

suffixes :: [a] -> [[a]]
suffixes list = list : case list of
  []   -> []
  _:xs -> suffixes xs
```

What is the time complexity of `suffixes`?

- $O(1)$
- $O(n)$
- $O(n^2)$
- $O(n^3)$
- $O(2^n)$

Save Answer

Q4.4

2 Points

Consider the following function:

```
-- | infixes
--
-- Returns the list of all possible contiguous sublists of the

infixes :: [a] -> [[a]]
infixes list = case list of
  []   -> [[]]
  x:xs -> map (x:) (prefixes xs) ++ infixes xs
```

What is the time complexity of `infixes`?

- $O(1)$
- $O(n)$

- ○ $O(n^2)$
- ○ $O(n^3)$
- ○ $O(2^n)$

Save Answer

Q5 Programming Questions
70 Points

Code templates for all programming questions in this exam can be found in the zip file here.

We recommend that you use VSCodium for your programming, and that you open the Terminal tool in VSCodium in order to test your code. Non compiling code can result in zero marks or lose significant marks. Please make sure each of your Haskell scripts compiles. **Do not change file names.** If you did not attempt or complete some functions, leave them `undefined` instead of commenting out the entire function definition. Gradescope will give you feedback once you upload each Haskell script on how many tests it passed. These tests may not be exhaustive and so passing all the tests may not guarantee full marks. You can overwrite the previous submission by uploading a new haskell file.

Using the incomplete template scripts, complete the undefined functions (marked with the comment `TODO`). The specification of each function is provided in the comments immediately above the function. You will need to adhere to that specification. You may use the doctests provided to help test your solutions. **These doctests may not be exhaustive**, and so passing all the available doctests may not guarantee full marks. We may also deduct marks for poor style.

Please submit by uploading the Haskell files.

You can find all programming questions on your dashboard as follows:

- Meal.hs [15 points]
- Differ.hs [10 points]
- Polymorphic.hs [10 points]
- HOF.hs [10 points]
- Filterable.hs [10 points]
- Trees.hs [15 points]

Save Answer

Save All Answers

Submit & View Submission