Australian National University

PROGRAMMING AS PROBLEM SOLVING (INCLUDING ADVANCED) COMPP1100/COMP1130

Important notice: Code templates for answers to coding questions in this exam can be found in the accompanying VSCodium project. You can place this Web page for the exam in a window side-by-side with the VSCodium project window, so that you can conveniently switch between them while working on your answers. You will also want to open the Terminal in VSCodium in which you can load your solution codes into GHCi and test them out. You can also make use of doctest for testing.

All of your answers will be auto-graded, so for coding problems you will be marked according to how many of our tests you pass. Incorrect answers that pass tests will be penalised accordingly. The auto-grader will only mark code that you upload to the exam by dragging it into the appropriate box from a file window.

Total marks: 50 (1100) or 60 (1130)
Reading period: 10 minutes
Writing period: 60 minutes
Permitted materials: One A4 page with notes on both sides, Unannotated paper-based dictionary.

Questions are not of equal value.
All questions must be completed on this web form.

Your work is automatically saved and recorded as you type.

This is a closed examination. You may not copy this exam.

Question 1 [20 Marks] *Multichoice Questions (1100 and 1130)*

These questions are for both COMP1100 and COMP1130 students. Each question is intended to have only one correct answer. Each question is worth 2.5 marks. An incorrect or missing answer is worth 0 marks, without further mark penalty.

1 i) [2.5 Marks] *Sets and Functions*

Let `𝔻` be the set `{North,South,East,West}` and `𝔹` be the Booleans `{True,False}` .

Which of the following defines a valid mathematical function `f` with domain `𝔻` and codomain `𝔹` ?

- [ ] `f(North)=True, f(South)=False`
- [ ] `f(North)=True, f(South)=False, f(True)=East, f(False)=West`
- [ ] `f(North)=True, f(South)=True`
- [x] `f(North)=True, f(South)=True, f(East)=True, f(West)=True`
- [ ] `f(North)=True, f(South)=True, f(North)=False, f(East)=False, f(West)=False`

Clear

1 ii) [2.5 Marks] *Sets and Functions*

Given any sets `A` , `B` , and `C` , and $\texttt{proj}_{\texttt{right}}$ the right projection of `A × B` , which of the following statements about the composition `g` . $\texttt{proj}_{\texttt{right}}$ is true?

- [ ] It is a valid mathematical function for any function `g :: A → C`
- [ ] It is a valid mathematical function for any function `g :: A × B → C`
- [x] It is a valid mathematical function for any function `g :: B → C`
- [ ] It is a valid mathematical function for any function `g :: C → A`
- [ ] It is a valid mathematical function for any function `g :: C → A × B`
- [ ] It is a valid mathematical function for any function `g :: C → B`

Clear

1 iii) [2.5 Marks] *Programming*

If a whole program, written in .hs files, has been converted into instructions that a machine can run directly, we say it has been

- [ ] Assembled
- [x] Compiled
- [ ] Evaluated
- [ ] Interpreted
- [ ] Mechanised

Clear

1 iv) [2.5 Marks] *Basic Types*

Which statement about `Int` and `Integer` is true?

- [x] Elements of `Int` are bounded in size, whereas elements of `Integer` are not bounded
- [ ] Elements of `Integer` are bounded in size, whereas elements of `Int` are not bounded
- [ ] `Int` is a valid Haskell type, whereas `Integer` is not a valid Haskell type
- [ ] `Integer` is a valid Haskell type, whereas `Int` is not a valid Haskell type
- [ ] `Int` and `Integer` are two different names for the same type

Clear

1 v) [2.5 Marks] *Basic Types*

Consider the function

```
baz :: Bool -> Bool -> Bool
baz b c = (not b && c) || (b == c)
```

Which of the following is a complete English language description of **all** cases for which `baz b c` will evaluate to `True` ?

- [ ] `b` and `c` have the same values
- [ ] `b` and `c` have different values
- [x] `b` is `False` or `c` is `True`
- [ ] `b` is `False` and `c` is `True`

Clear

1 vi) [2.5 Marks] *Algebraic Datatypes*

Consider the datatype definition

```
data Qux = Rax Bool Bool Bool | Tix Bool | Vox Bool Bool
```

How many different values are elements of this datatype?

- [ ] 12
- [x] 14
- [ ] 32
- [ ] 48
- [ ] 64

Clear

1 vii) [2.5 Marks] *Recursion with Lists*

Consider the function

```
dividing :: [Double] -> Double
dividing list = dHelper 1.0 list
  where
    dHelper :: Double -> [Double] -> Double
    dHelper x list = case list of
      [] -> x
      y:ys -> dHelper (y / x) ys
```

When applied to the list `[2.5,5.5]` , what mathematical calculation is computed?

- [ ] `(2.5 / 1.0) / 5.5`
- [ ] `(2.5 / 5.5) / 1.0`
- [ ] `(5.5 / 2.5) / 1.0`
- [ ] `(5.5 / 1.0) / 2.5`
- [ ] `2.5 / (5.5 / 1.0)`
- [x] `5.5 / (2.5 / 1.0)`

Clear

1 viii) [2.5 Marks] *Parametric Polymorphism*

Suppose we have the parametric polymorphic datatype definition

```
data Foo a b = Bar a b
```

Which of the following is a valid **element of** an instantiation of this type?

- [x] `Bar 3 "hello"`
- [ ] `Bar Int String`
- [ ] `Foo 3 "hello"`
- [ ] `Foo Int String`

Clear

Question 2 [10 Marks] *Multichoice Questions (1130 ONLY)*

These questions should be attempted **only** by COMP1130 students. Responses by COMP1100 students will be ignored. Instructions are otherwise as for the previous questions.

2 i) [2.5 Marks] *Lambda Calculus*

What are the free variables of the following lambda-calculus term?

```
λx.((λy. y z) y)(x z)
```

- [ ] The empty set
- [ ] `{x}`
- [ ] `{y}`
- [ ] `{z}`
- [ ] `{x,y}`
- [ ] `{x,z}`
- [ ] `{y,z}`
- [ ] `{x,y,z}`

Clear

2 ii) [2.5 Marks] *Lambda Calculus*

Consider the term

```
(λx. x (λy. x y) (λx. z x)) (y y)
```

Which one of the following lambda-calculus terms is (alpha-equivalent to) the result of **one** step of beta-reduction to this term?

- [ ] `(λx. x x (λx. z x)) (y y)`
- [ ] `(λx. x (x (λx. z x))) (y y)`
- [ ] `λx. y y (λy. x y) (λx. z x)`
- [ ] `(λy. x y) (λx. z x) (y y)`
- [ ] `y y (λw. y y w) (λx. z x)`
- [ ] `y y (λw. y y w) (λx. z (y y))`
- [ ] `y y (λy. y y y) (λx. z x)`
- [ ] `y y (λy. y y y) (λx. z (y y))`

Clear

2 iii) [2.5 Marks] *Lambda Calculus*

Assuming we have encodings of Booleans, consider the Barendregt encoding of natural numbers:

```
0   as λx. x
n+1 as λx. if x then False else n
```

Now consider the lambda term

```
λx. if (x True) then False else (x False True)
```

On which inputs of (Barendregt encodings of) natural numbers will this evaluate to `True` ?

- [ ] 0 only
- [ ] every number except 0
- [ ] 1 only
- [ ] every number except 1
- [ ] 0 and 1 only
- [ ] every number except 0 and 1
- [ ] every number
- [ ] no numbers

Clear

2 iv) [2.5 Marks] *Lambda Calculus*

What is the complete beta-reduction of

```
(λw. w ((λx. x) w))(λy. y z)
```

according to **strict** evaluation?

- [ ] `→ (λw. w w)(λy.y z) → (λw. w w) z → z z`
- [ ] `→ (λw. w w)(λy.y z) → (λy.y z)(λy.y z) → (λy.y z) z → z z`
- [ ] `→ (λw. w ((λx. x) w)) z → (λw. w w) z → z z`
- [ ] `→ (λw. w ((λx. x) w)) z → z ((λx. x) z) → z z`
- [ ] `→ (λy.y z)((λx. x)(λy.y z)) → (λx. x)(λy.y z) z → (λx. x) z z → z z`
- [ ] `→ (λy.y z)((λx. x)(λy.y z)) → (λx. x)(λy.y z) z → (λy.y z) z → z z`
- [ ] `→ (λy.y z)((λx. x)(λy.y z)) → (λy.y z)(λy.y z) → (λy.y z) z → z z`

Clear

Question 3 [30 Marks] *Programming Questions (1100 and 1130)*

There are **six** Haskell files that you need to complete and submit. Each file contains exactly one function to complete, although you may define helper functions if you wish. You may use any Haskell function available in the Prelude or basic libraries.

You will find the template Haskell files in a folder on your desktop, and in VSCodium.

Please submit by dragging and dropping **each** Haskell file into the white box below **each** question. Do not rename the files before submission. You should be able to see automatic test results for your files after submission. The doctests in the file are intended to help you, but are not identical to the tests that will be used to mark you, and are not intended to be exhaustive. You may submit to the same question multiple times; if you do so, your previous submission will be overwritten.

These questions are auto-graded: you will be graded according to how many tests you pass. The auto-grader will only mark what you upload to the exam. Therefore it is **essential** that you upload your code into this exam. Note that **code that cannot run** will receive zero marks, even if part of the code is correct. In particular, if you import packages that are not basic libraries, your code will receive zero marks.

3 i) [6 Marks] *AnyNeg.hs*

Drag and drop AnyNeg.hs below
AnyNeg.hs 2023-08-26 12:24:51

```
module AnyNeg where

-- | anyNeg:
-- Given a product of three Ints (i.e. a triple) as input,
-- return True if any of the Ints are strictly less than zero.
-- Otherwise, return False.
--
-- Examples:
--
-- >>> anyNeg (2,0,-4)
-- True
--
-- >>> anyNeg (-2,0,-4)
-- True
--
-- >>> anyNeg (2,0,4)
-- False
```

3 ii) [6 Marks] *SafeIntDiv.hs*

Drag and drop SafeIntDiv.hs below
SafeIntDiv.hs 2023-08-26 12:32:25

```
module SafeIntDiv where

-- | safeIntDiv:
-- Given two Ints as input,
-- return a Maybe Int as output according to the following:
-- - if the second number is zero, return Nothing
-- - if the first number is not divisible by the second, return Nothing
-- - otherwise return the value of the first number divided by the second.
--
-- Hint: the Prelude provides two functions that may be useful
-- div :: Int -> Int -> Int divides two Ints, discarding the remainder;
-- rem :: Int -> Int -> Int gives the remainder of the division of two Ints.
--
-- Examples:
--
-- >>> safeIntDiv 8 0
-- Nothing
```

3 iii) [6 Marks] *DietaryMessage.hs*

Drag and drop DietaryMessage.hs below
DietaryMessage.hs 2023-08-26 12:37:28

```
module DietaryMessage where

-- The DietaryRequirement type lists some possible Dietary Requirements.
-- In the case of an Allergy, a String argument records the specific food.
-- DO NOT edit or delete this type declaration.

data DietaryRequirement
  = Halal
  | Kosher
  | Vegan
  | Vegetarian
  | Allergy String

-- | dietaryMessage:
-- Given an input of type DietaryRequirement,
-- return a String communicating that requirement,
-- *exactly* according to the following specification:
```

3 iv) [4 Marks] *Oblong.hs*

Drag and drop Oblong.hs below
Oblong.hs 2023-08-26 12:45:39

```
module Oblong where

-- | oblong:
-- The 'oblong numbers' are the sequence 0,2,6,12,20,30,...
-- where the gap between each pair of numbers increases by 2 each time.
--
-- Write a function that, given a non-negative Int n as input,
-- returns the nth oblong number
-- (where their index starts at zero)
-- i.e. send 0 to 0,
--           1 to 2,
--           2 to 6, and so on.
-- If the input is negative, return 0.
--
-- Examples:
--
-- >>> oblong (-3)
```

3 v) [4 Marks] *ReplaceSecond.hs*

Drag and drop ReplaceSecond.hs below
ReplaceSecond.hs 2023-08-26 16:26:16

```
module ReplaceSecond where

-- | replaceSecond:
-- Given a Double and a list of Doubles as input,
-- return the input list unchanged, except that
-- the second element of the input list is replaced by the input Double.
-- If the list has fewer than two elements, return the list unchanged.
--
-- Examples:
--
-- >>> replaceSecond 1.1 [0.0]
-- [0.0]
--
-- >>> replaceSecond 1.1 [0.0,2.2,3.3,4.4]
-- [0.0,1.1,3.3,4.4]

replaceSecond :: Double -> [Double] -> [Double]
```

3 vi) [4 Marks] *Hexadecimals.hs*

Drag and drop Hexadecimals.hs below
Hexadecimals.hs 2023-08-26 16:49:59

```
module Hexadecimals where

-- DO NOT edit or delete this library import
import Data.Char

-- | hexadecimals:
-- Given a list of Ints as input,
-- return a String (list of Chars),
-- with each Int between 0 and 15 inclusive
-- replaced by its hexadecimal character.
--
-- (the hexadecimal characters are '0','1','2',...,'9','a','b',...,'f',
-- and are used to represent base 16 digits).
--
-- To achieve this, use a function in the Data.Char library with name and type
-- intToDigit :: Int -> Char
-- which converts Ints between 0 and 15 inclusive to hexadecimal characters,
```