Slide 1

Format of the Exam

Instructions:

- Time: 3 hours
- 4 Questions (with 32 subquestions)
- Answer each in a seperate answer booklet and one on multiple choice answer sheet
- Do not write answers into the examination paper
- Pencil for multiple choice answer sheet & ink for answer books
- Closed book
- 5% penalty for not following the examination instructions

Time and Venue:

`https://my.unsw.edu.au/student/academiclife/assessment/examinations/COMP.html`

Slide 2

The Four Questions

Question I (Multiple Choice):

- 20 subquestions (20/60 marks)
- Answer with pencil on generalised answer sheet
- Marking:
  - Correct answer: 1 mark
  - Incorrect, multiple, or wrong answer: 0 marks
- Example of question format:

  Given a function of type

  ```
  foo :: [a] -> [b] -> [(a, b)]
  ```

  which of the following applications of `foo` result in a type error?

  (A) `foo [] []`
  (B) `foo [1, 2] ["Hello", "World"]`
  (C) `foo [1, 2] [3, 4]`
  (D) `foo [1, 2] [True]`
  (E) None of the above; i.e., all applications are type correct

  Answer: E

Slide 3

Question II (Short Answer):

- 10 subquestions (20/60 marks)
- Answer with ink in an answer booklet
- Do not answer on the question paper
- Example of question format:

  The function `length` is supposed to compute the length of the argument list. Complete its definition.

  ```
  length        :: [a] -> Int
  length []      = 0
  length (x:xs)  = 1 + [          ]
  ```

  Example answer: `length xs`

Slide 4

Question III (Programming):

- 1 subquestion (10/60 marks)
- Answer in an extra answer booklet
- Try to provide clear and easy to understand answers
- Don't forget the type signatures
- Confusing and illegible answers will lose marks
- You may elect that this question is not marked ⇒ record 1 mark for the question

Question IV (Programming/Theory):

- 1 subquestion (10/60 marks)
- Answer in an extra answer booklet
- Answers as in Q III
- You may elect that this question is not marked ⇒ record 1 mark for the question
- Partial solutions to this question (i.e., worth less than 50%) will score 0 marks in this question

Slide 5

Strategic Advise

- ➜ Do the easy questions first
- ➜ Don't get stuck in a question
- ➜ On the question paper, keep track of which questions you have done already

Slide 6

Summary of Material Covered by the Exam

Slide 7

Basic Programming Concepts

- ➜ Function definitions
- ➜ Types & type signatures
- ➜ Recursion
- ➜ Pattern matching & guards
- ➜ Lists & tuples (also lists of lists and lists of tuples etc.)
- ➜ Basic Haskell functions: `length`, `++`, `sum`, `head`, `tail`, `init`, `last`, and so on

Slide 8

Unix Literacy

- ➜ Basic concepts:
  - What does an operating system do?
  - What is a file?
  - What is a process?
  - What are file permissions?
  - What is an interpreter? What is a compiler?
- ➜ Basic commands, such as `cd`, `ls`, `chmod`, and `grep`—you should also know the most common options of those commands, e.g., `ls -l`
- ➜ Pipes (|) and redirections (<, >, >>)
- ➜ Basics of shell scripts

Unix Literacy: File Permissions

Slide 9

The Unix kernel ensures safety and & security:

- Each file is owned by a user
- Each file is—in addition—associated with a group of users
- Each file has access permissions

The command `ls -l` prints permissions in the following style:

d $\underbrace{\texttt{rwx}}_{\text{user}}$ $\underbrace{\texttt{r-x}}_{\text{group}}$ $\underbrace{\texttt{r-x}}_{\text{other}}$

- `r`ead, `w`rite, and e`x`ecute
- Execute permissions for directories are special
- Permissions can be altered with `chmod`—check the online documentation with `info chmod`.
- The command `info info` describes the online documentation system

Slide 10

Examples from the lecture notes:

What does the following do?

```
grep cost *.hs >result
```

And this?

```
grep PriceList *.hs >>result
```

How about this?

```
grep wed tute-allocations | less
```

And this?

```
grep wed tute-allocations | grep tuba
```


More Advanced Programming Concepts

Slide 11

Advanced List Processing:

- Association lists (lookup, updating, etc.)
- Try to write a function

  ```
  sublists :: [a] -> [[a]]
  ```

  that computes all sublists of a given list.
- Searching & sorting:
  - Divide-and-Conquer
  - Sorting algorithms
  - Complexity of sorting

Data Types:

- How does a `data` definition work?
- What is an enumeration type?
- When is the `Maybe a` type useful?

Slide 12

Trees:

- What is a tree? What is an augmented tree? What is a binary tree? What is a path?
- Functions processing trees: creation of a tree & tree walks (e.g., how to compute the number of nodes in a tree, or collect all leaf values in a list)
- What is an ordered tree? How can you write a function constructing an ordered tree? What is the height of an ordered tree? How can you find a value efficiently in an ordered tree?

Higher-Order Functions:

- What is a higher-order function?
- How do `map` and `filter` work and how can you use them?
- Try to write your own sorting function, which gets a comparison function as an argument—i.e., a function of type

  ```
  sortBy :: (a -> a -> Bool) -> [a] -> [a]
  ```

Slide 13

Modules & Decomposition:

- What is a module? What is an import list and what an export list? What are local definitions?
- Why is program decomposition important? Why are modules important?
- In which ways can a program be decomposed?
- How does a good design look like? What is cohesion? What is coupling?

Input/Output:

- What is an action? What is the type of an I/O action?
- Basic I/O functions like `putStr`, `getLine`, and so on
- How can I/O actions be combined?
- Reading from and writing to a file

Slide 14

Formal Reasoning:

- What is equational reasoning and how does it work?
- What is induction and how does it work?
- Induction over numbers and structural induction over lists
- What is a lemma and how can you use it?
- You should be able to do proofs as those in the tute and lab questions

Work Complexity:

- How do you construct a timing function?
- What is the $\mathcal{O}$-notation and how is it used?
- What is a witness in this context?
- What is a linear algorithm? What is a logarithmic algorithm? What is a quadratic algorithm?

Slide 15

Good Luck!