COMP1140/1110/6710
Structured programming

Revision – End of Semester

The final exam

* is scheduled 9.30am AEST on Tuesday the 8th of November;
* is 3 hours;
* is on-line: access/submit through exam gitlab;
* is open-book: you may refer to course material, notes, documentation, search the web, etc;
* is individual: you must not communicate with anyone other than the course staff through private posts on piazza.

* This week
  - Ensure you have a working IDE
  - Clone the exam repository to the computer you will be using for the exam.
  - Access to the repo will close on Friday.
  - Setup and test self-invigilation (optional).
* During the exam
  - At the start, pull updates from your exam repository – it will now have questions in it.
  - push your work as you complete it.

https://comp.anu.edu.au/courses/comp1110/assessments/campus_only/final/

Self-invigilation

* You record yourself during the exam.
* You keep the recording.
* If we find cause to doubt the integrity of your exam submission, then you may submit your recording as evidence in your favour.
* For a recording to be useful, it must:
  - show your entire screen (at readable resolution);
  - identify you, with audible sound;
  - for the entire duration of the exam.

Core knowledge

* Basic programming *in Java*:
  - design iterative/recursive algorithms;
  - variable scope & state, references;
  - effective use of standard library.
* Object-oriented programming *in Java*:
  - classes & inheritance, polymorphism, nested classes;
  - object creation & initialisation, equality, hashing, comparison.
* Data structures:
  - interfaces and implementation design choices;
  - time and space complexity.
* Software development practice:
  - writing *good* tests (using JUnit);
  - effective use of tools (IDE, git).

Revision topics

Recursion

* Recursion is a way of thinking about how to solve computational problems.
  - "If I had the answer to problem(sub) ...', then I could easily find the answer to problem(this)".
  - Example: The height of a (binary) tree is the max depth of any node (longest path from root to any leaf node). If the height of the left subtree is h_L and the height of the right subtree is h_R, then the height of this tree is $1 + \max(h_L, h_R)$.
* Straightforward mapping from recursive *idea* to recursive implementation.
  - Tricky bit is often how to collect the results.

What makes good test cases?

* Satisfy assumptions/restrictions.
* Simple (enough that correctness can be determined “by hand”).
* Cover the space of inputs *and* outputs.
* Cover branches in the code.
* Cover “edge cases”. For example:
  - What if the string/collection is empty?
  - What if *n* is zero?
  - Any value that requires special treatment in the code.

`static`

* Declare field, method or nested class `static`.
* `static` is the opposite of dynamic:
  - method dispatch by type of the referring variable, not the object;
  - field reference is determined statically.
* Static context: no `this`.
* `import static`.