The Exam

A brief look backwards

- This course roughly covers three previous modules
  - Software Engineering
  - Algorithms and Data Structures
  - Object Oriented Programming

  (though there are things we miss out, plus others we add)
- Previously two modes of assessment were used
  - Take-home courseworks over 1 week+
  - Paper and pen examinations

Previous paper exam question

f) Figure 1.3 shows the type declaration for a dynamic linked list, where each node stores an integer in the *data* field.

```
struct Node {
    int data;
    Node * next;
};

typedef Node * NodePtr;
NodePtr hdList = NULL;
```

Figure 1.3 Linked list declaration.

i) Write a C++ function/procedure that takes as its argument a pointer to the linked list and returns a pointer that points to the last node of the list. If the list is empty, your function/procedure should return the NULL value. [3]

ii) Write a C++ function/procedure that takes as argument a pointer to the linked list, and sets all values stored in the *data* field of the nodes to the zero value, except from the *data* field of the last node. [4]

Criticism of previous approaches

- Take-home courseworks
  - **Time** : students sink unlimited time into them
  - **Competition** : students feel they compete with each other
- Written paper examinations
  - **Results** : over-focus on small functions, not problem solving
  - **Format** : writing C++ code using a pen is unrealistic
  - **Practise** : modern coding relies on documentation

What we are trying to achieve

- Mid-terms: *individual assessment of knowledge*
  - Tested your knowledge and conceptual understanding

- 1-day take-home exam : *ability to deliver solutions*
  - **1-day** : limits time spent, but enough for real problems
  - **Take-home** : can use standard methods and tools
  - **Pair-work** : demonstrate and test collaborative working

Doing something useful

- The exam questions are supposed to be "useful"
  - They perform a task that might encounter in practise
  - They are connected to some other aspect of EEE
    - Programming is a generic tool, like maths
    - It has little value by itself
- Useful work requires some context or background
  - *What is the extra information needed?*
  - *What existing decisions have been made?*
  - You may find yourself spending a lot of time reading

Exam process

Exam Timing

Exam date: Mon 4th April

10:00 : Exam questions released
11:00 : Answers to any queries shared with all
During : Push to repo at least every 60 mins
18:00 : Final submission

You can be anywhere during this period

...you just need the internet

Exam Timing

10:00 **: Exam questions released**

11:00 : Answers to any queries shared with all

During : Push to repo at least every 60 mins

18:00 : Final submission

Exam format

- The exam is "answer two out of three"
  - You are given three questions
  - You ***choose*** two that you wish to be assessed
- Everyone gets the same questions
  - No per-team modifications or differences

Mock questions are in the same format as the final

Exam Timing : Release of Questions

- Questions will be released via three routes
  - Repository made public in github: https://github.com/ELEC40004/elec40004-2019-exam
  - Archive (.tar.gz) of repo posted to blackboard
  - Archive (.tar.gz) of repository emailed via college email
  - As far as possible these will be simultaneous
- The exam repo is also the submission repo
  - Clone the repo
  - Modify and add files
  - Push and submit your modified version

Exam Timing

10:00 : Exam questions released

11:00 : Answers to any queries shared with all

During : Push to repo at least every 60 mins

18:00 : Final submission

Exam timing : queries

- There may be questions arising about the exam
  - Just like in a normal exam
  - e.g. bug in the question, unclear instruction, ...
- In the first hour queries can be submitted by email
  - Queries will be collected and considered/checked
- At (about) 11:00 a response to queries is made:
  1. "The exam is correct as written"; or
  2. An amendment will be shared with all students
- The preference is not to change the questions
  - Make changes to marking rather than change spec
  - Just like in a normal exam

Exam Timing

10:00 : Exam questions released

11:00 : Answers to any queries shared with all

During **: Push to repo at least every 60 mins**

18:00 : Final submission

Exam timing : intermediate pushes

- Teams must push to shared repo every 60 minutes
  - These pushes are needed to form a record of your work
  - 60 minutes is a minimum: you can push more frequently
- There must be a clear organic history
  - *How did the solution evolve over time?*
  - *Who committed what and when?*
- Both team members are expected to have commits
  - You can update a text log file if there is no new code
  - e.g.: briefly describe your thought processes or activity
- You can/should push work-in-progress on branches
  - Master does not have to change

Exam Timing

10:00 : Exam questions released

11:00 : Answers to any queries shared with all

During : Push to repo at least every 60 mins

18:00 : Final submission

Exam timing : submission

- Submission is via three routes
  - *Primary*: your team's **master** branch in github
  - *Secondary*: a hash submitted via blackboard
  - *Tertiary*: a hash submitted via Microsoft online form
- Secondary and tertiary are proof of existence
  - A specific commit existed ***before*** the deadline
  - You must be able to produce the associated repo
- These are mainly in case of temporary disruption
  - *What happens if github goes down?*
  - *What happens if pushing is taking too long?*
  - *What happens if our local wifi suddenly breaks?*

Exam timing : submission

- Your team must explicitly nominate two questions
  - The root readme file has three check-boxes
  - Your commit should "tick" two of them
  - Can work on all three questions, then choose at the end
  - Exactly the same idea as a normal 2 of 3 exam

- If you don't pick two, then we assume q1 and q2
  - There is no max function over the three questions

Marking : process

- All marks are quantitative and functional
  - *Does your repository contain the required things?*
  - *Does your repository perform the correct functions?*
  - Marking is primarily by scripts
    - Completely repeatable
    - Completely objective
- The only thing that matters is meeting the spec.
  - No points for style or code quality
  - No marks added/removed for clever/poor code

Marking : errors and mistakes

- Solutions need to be "perfect" to get 100%
  - Just like in a normal exam
- Some manual fixes are allowed, with a penalty
  - Mis-named files/functions/classes
  - Compile errors due to not testing on the right platform
  - Difficult to anticipate crashes *may* be fixed
- Other manual fixes will not be applied
  - Code that could *never* compile will get zero
  - Code that always crashes will get zero
  - There are no marks for producing code sketches

It's better to have a small number of correct answers than a large collection of broken answers

Marking : pair-work

- Teams are marked as a single unit
  - It is expected that both members get the same mark
  - Individual understanding was tested in mid-terms
- Team members may contribute different amounts
  - Real-life teams have different levels of individual ability
  - The team should work to maximise overall output
- Both team members must contribute *something*
  - The git history should contain evidence of both members
- In *exceptional* cases different marks may be given
  - This would require real dysfunction, with evidence that one member was given the chance to contribute and didn't

Mock assessment days

- We plan to hold two "mock" assessment days
  - Follow the timing and structure of the exam day
  - Release an exam repo at 10:00
  - Collect repo snapshots during the day
  - Test our various processes work
- You can also take part as practice for timing
  - Develop your co-working skills
  - Work out how to collaborate and manage things
  - Practise questions under exam conditions
- Tentative dates
  - Mon 30th March
  - Mon 20th April
  - Same set of questions used for both

Coronavirus : remote working

- No changes are needed due to remote teaching
  - You can work anywhere

Plagiarism

Plagiarism : expectations

- **Code**: All submitted code is written by your team
  - No code copied from external web-sites
  - No code from other teams
  - No code from friends/family/others
- **Solutions**: your submission represents *your* thinking
  - No brain-storming with other teams
  - Don't ask questions on programming web-sites
  - Don't consult with friends/family/others
  - Avoid giving hints about approaches to other teams

Plagiarism : proving it's your work

- The 60 minute push rule is to "show your working"
  - *How was the code developed?*
  - *In what order was it completed?*
  - *Who wrote which parts?*
  - *What were you thinking about when not coding?*
- Each teams development process will be unique
  - Your solutions may end up somewhat similar
  - Your code will definitely be different
  - Your bugs/fixes will definitely be different
- Your commit history is evidence that you did the work

Plagiarism : detection

- A number of tools will be used to look for plagiarism
  - Statistical analysis of commit histories
  - Fuzzy code analysis tools to look for similarities
  - Code analysis tools to look for large jumps in functionality
  - Bug analysis scripts to look for common failure modes
- Any flagged outliers will be investigated manually
  1. Manual inspection of code to eliminate false positives
  2. Oral questioning of teams to query anything odd
  3. Report suspected cases of plagiarism to dept. committee

We do not want or expect this process to find anything

Plagiarism : working practice

- During the day you can operate normally
  - Talk to who-ever you want to
  - Work wherever you want
  - Use web-sites and email as normal
  - Google things; use stack-overflow; ...
- Just avoid sharing trade secrets with each other
  - Imagine you are competing companies in sharing a collaborative working space
  - People at Intel know people at AMD
    - They socialise at conferences, go for lunch, talk about stuff
    - They don't share details on unreleased chip designs

Forming teams

Team selection

- You have two choices:
  1. Form a pair and tell us (both members should submit): https://forms.office.com/Pages/ResponsePage.aspx?id=B3WJK4zudUWDC0-CZ8PTB07FeicCPhVAsezU-PfpjRxURUlSRFBXM1M4S0FMQklQRDRDTzNJTkg5WS4u
  2. Wait and be assigned a team
- Team selection timelines
  - Mon 23rd Mar : deadline for self-selecting a team
  - Fri 27th Mar : notification for final teams
    - Shared repos created and distributed

Team preparation

- You need to practice working with your partner
  - How do you communicate?
  - How do you split tasks up?
  - What are your relative skills?
- You have a number of approaches you could take
  - Pair-programming : two people work on one problem
  - Parallel tasks : work together on independent problems
  - Dev. vs. test : one person writes code, other on writes tests
- How you work and where you work is up to you