Projects

This course is examined via three projects, with topics "something I learnt in $\geq$ 1st year" (20%), "something I learnt in $\geq$ 2nd year" (30%) and "something I learnt this year" (50%). Please choose better titles than that!

Deadlines: end of week 4, end of week 8, end of week 12. I believe it's always 1pm

If you're late then this is not a matter for me, you need to talk to the senior tutor

For the first two projects, I will try and turn the marking around within a few working days, and I will give you as much feedback as I can

“What do I do for my project?”

I want *you* to decide what to do for your project. If I make one suggestion to one of you, then suddenly ten of you want me to make suggestions.

If you want some inspiration for your first project: go and look at your first year notes.

Or your first year problem sheets.

Or an old first year exam.

Any of these are fine.

But feel free to run ideas past me.

“Am I allowed to do X?”

Yes, as long as you learnt it in the maths department at some point.

If you learnt it in another department, the answer is “probably – ask me”.

Are you allowed to use theorems from Lean’s maths library? Yes!

Can you reprove theorems which are already in Lean’s maths library? Yes!

Is your project feasible?

Is your idea really really hard to do in Lean? It might be.

You might want to find that out now.

Or you can find out while you're doing it, and *this is OK*.

Just explain why things were much harder than expected in your write-up – *this is fine*.

Some stuff which is harder than you might think.

1) Finiteness.

Why? Because we have a good intuitive grasp of it, and don't think about how hard formalising it is.

What is the definition of the size of a finite set? How do we know every finite set has a size? How do we know a finite set can't have two different sizes?

Lean has a lot of results about finiteness in its maths library.

Lean can *do* finiteness, it's just that there's a "knack" to it.

Some stuff which is harder than you might think.

2) Calculations.

Lean's maths library was primarily designed to prove theorems, not to do real number calculations.

It's getting better, but you have been warned.

Tricky example: prove $0.54 < \cos(1) < 0.55$ (1 radian, not one degree).

Plagiarism

I find it quite sad that I have to worry about marks and plagiarism and so on.

What I want to do is to *teach you Lean*.

If you are stuck and you ask on Ed Stem or the Discord or the Zulip how to do it and someone tells you, *this is not plagiarism*.

Just write in the comments or in the pdf "some random user called `xyzabc` on the Discord showed me this trick".

You can ask your friend. You can even thank your friend in your pdf!

Plagiarism is *passing off someone else's ideas as your own*.

What is the mark scheme?

Here is the thought experiment.

Consider an external examiner (a professor from another UK university).

They have heard of Lean and that know it's a "theorem prover". They have no idea how it works.

What would they think of your project?

Would they be able to *understand something about what you are doing?*

Would they think "this is pretty cool, we should be teaching this at my university"?

What is the mark scheme?

For the first project. I am looking for:

- some Lean code (150–200 lines of code including comments? More if you like?)
- A pdf explaining what is going on (5 pages? More if you like?)

You won't get penalised for doing too much (but it will take longer).

I'm expecting about 15 (or more) hours of work.

Note that working on Lean by yourself is harder than doing my problem sheets!

What is the mark scheme?

Things which I *do not want in the pdf*:

1) 5 pages explaining what Lean is. Assume everyone reading it knows what Lean is.

2) 5 pages explaining what certain tactics do. Assume everyone reading it can look this up.

What is the mark scheme?

Things I would be happy to see in the Lean file/files:

1) Explanations of what is happening (i.e., comments in the code)

2) No errors – but correct theorems with sorried proofs are OK. [More dangerous: sorried definitions, false theorems with sorried proofs].

3) Correctly-formatted code! (see this part of the notes), or read the mathlib style guide).

What is the mark scheme?

Things I would be happy to see in the pdf file:

1) Explanations of what is happening in the code;

2) Discussion of things you found hard or surprising;

3) Comments about what you have learnt.

What is the mark scheme?

Each project will be marked out of 100.

30 points for presentation;

30 points for content;

40 points for “bonus”.

For the first project, there will also be 0 points for the compulsary 10 minute oral exam.

For the final project there will be 0 points for the compulsary 3 minute project presentation which you’ll give on Teams between 2 and 4pm on the Friday of week 12.

What is the mark scheme?

Presentation: does the pdf look nice? No typos? Does it clearly explain what it's doing? Would a 1st year understand it if they'd been taught the material?

Does the Lean code have comments when the going gets tough? Is the code correctly formatted, with the right indentation, and "dots" being used appropriately when there are two or more goals?

What is the mark scheme?

Content: Is the material at an appropriate level? Does the Lean code compile? Does it *lint*? (try `#lint` at the end of your file!)

Do the definitions have docstrings? Do the main theorems have docstrings? Do you explain what you're doing in the pdf, in a way which an external examiner would understand?

What is the mark scheme?

Bonus: At my discretion.

Ideas: Have you done something unexpected? Something difficult?

Have you done something in a new way?

Do the code and the write-up “feel professional”?

This stuff is closer to cutting edge modern research than you might guess!

How to start on a project

The easy way: just make a new directory in the course repository called `my_first_project` and get going. I would recommend that way.

Make sure you keep a backup.

Alternative approach: make a new project with `lake` (and then remember to submit your `lakefile.lean` when you submit). Instructions: here

Advice

- Start early
- Ask for help early!
- Make sure you *completely* understand the informal proof before formalising your proof (if you are stuck on what tactic to use next in Lean, the first question you will be asked is, what's the informal proof)
- Use lots of `have` statements, and having lots of small lemmas is much better than having one very long proof

How to hand in a project

There will be a turnitin drop box on the blackboard site.

I *think* that you can only upload *one file*. But your project will be several files! (a Lean file and a pdf file, at least).

So you need to “zip up” all the files you want to hand in, using a standard program.

Instructions on how to zip up all your files into one big file to hand in, will also be on blackboard.

What we'll be doing for the next couple of lectures

- You have almost all the basic tactics to write many proofs: see `https://b-mehta.github.io/formalising-mathematics-notes/Part_2/Part_2.html`
- Next we'll talk about more complex declarations, typeclasses, and structures
- And then we'll talk about how Lean/Mathlib does topology, graph theory, number theory, etc
- I am trying to do material which *helps you to do your projects*
- As you might be beginning to understand, you don't have to do the course repo sheets in the right order, or indeed at all. But they'll teach you something

Questions

Any questions?