V3A6 Mock exam

Skills needed

- Write a short Lean proof (if all needed Mathlib lemmas are provided).
- Understand in type theory of Lean what terms have what types.
- Know commonly-used Lean tactics.
- Be able to state definitions and theorems in Lean in various areas of Mathematics.
- Be able to spot mistakes in definitions, statements or proofs.
- Be able to know what is wrong based on an error message.

Skills not tested on the exam

- You do not need to know Mathlib lemmas
- No questions will be asked about measure theory, implementing logic, implementing (graph) algorithms, category theory, differential geometry, metaprogramming, cardinals, ordinals.

Instructions

- In many answers, you are asked to write a Lean proof. Write this Lean proof on paper
- You do not need to write a starting `by`. If you are asked to write a Lean proof of

  ```
  example {p : Prop} : p → p
  ```

  then a valid answer would be, e.g.

  ```
  intro h
  exact h
  ```

- If you are not asked to prove something a particular way, any valid Lean proof is accepted.
- If you are asked to write something using **basic tactics**, you should not use tactics that can perform many steps at once, such as `grind`, `simp`, `ring`, `tauto` or `linarith`. Tactics like `exact?` are also not allowed in such proofs.

Mathlib lemmas

Here is a list of some definitions and lemmas from mathlib which you may or may not need for the exam.

- `Prod.mk.{u, v} {X : Type u} {Y : Type v} (fst : X) (snd : Y) : X × Y`
- `add_assoc.{u_1} {G : Type u_1} [AddSemigroup G] (a b c : G) : a + b + c = a + (b + c)`
- `add_comm.{u_1} {G : Type u_1} [AddCommMagma G] (a b : G) : a + b = b + a`
- `add_zero.{u} {M : Type u} [AddZeroClass M] (a : M) : a + 0 = a`
- `zero_add.{u} {M : Type u} [AddZeroClass M] (a : M) : 0 + a = a`
- `one_mul.{u} {M : Type u} [MulOneClass M] (a : M) : 1 * a = a`
- `mul_one.{u} {M : Type u} [MulOneClass M] (a : M) : a * 1 = a`

Exercises:

1. Using only **basic tactics**, give a Lean proof for

   (a) `example {p q : Prop} : p → q → p`

   (b) `example {p q r s : Prop} (hp : p → r) (hq : q → s) : p ∧ q → r ∧ s`

   (c) `example {p q : Prop} : p ∨ q → q ∨ p`

2. The following are all proofs that can be done with a single tactic. Give a Lean proof for these examples (you are allowed to give a longer proof).

   (a) `example {x y : ℝ} (hx : 0 ≤ x) (h : 5 * x ≤ 2 * y - 1) : x ≤ y / 2`

   (b) `example : Differentiable ℝ (fun x : ℝ ↦ exp (x ^ 2) + sin (π * x))`

   (c) `example {p q : ℝ} (h : p = q) : p = q + 0`

   (d) `example {x y z : ℝ} : x * y * z * (x + y) * (x - y) = x * y * z * (x ^ 2 - y ^ 2)`

   (e) `example {x y : ℝ} (hx : y ≠ 0) : x * y / y = x`

   (f) `example {a b c d e : ℝ} (hab : a ≤ b) (hc : 0 ≤ c) (hde : d ≤ e) : exp a * c + d ≤ exp b * c + e`

   (g) `example {p : Prop} : p ↔ p ∧ True ∨ False`

3. Using the lemmas

   ```
   exp_add (x y : ℝ) : rexp (x + y) = rexp x * rexp y
   exp_mul (x y : ℝ) : rexp (x * y) = rexp x ^ y
   ```

   give a `calc`-proof of
   `example {x y z : ℝ} : exp (z * x + z * y) = (exp x * exp y) ^ z.`

4. State the following things in Lean.

   (a) (i) The real numbers are an abelian group (under addition).

   (ii) The non-zero real numbers $\mathbb{R}^* = \{x \in \mathbb{R} \mid x \neq 0\}$ are an abelian group (under multiplication).

   (iii) For all n, we have $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$.

   (iv) The real number 2 is the least upper bound of the interval $(-\infty, 2)$.

   (b) Which of the statements in part (a) could be made an instance?

5. Give a definition of a function $f : \mathbb{N} \to \mathbb{N}$ such that for $n \geq 3$ we have

$$f(n) = f(n-1) + f(n-2) + f(n-3)$$

   and $f(1) = f(2) = 1$ and $f(4) = 4$. (You can define your function differently than this characterization and do *not* have to prove that your function satisfies these properties)

6. In the following exercise, assume your local context includes the hypotheses `{X Y Z : Type} {f : X × Y → Z} {a : X} {b : Y} {c : Z}`. For each of the following expressions, determine if it is well-typed. If not, explain why. If it is well-typed, give its type.

   (a) `f a`

   (b) `(c, f (b, b))`

   (c) `(c, f (a, b))`

   (d) `Prod.mk X Y a b`

   (e) `Prod.mk (Y := X) (X := Y)`

   (f) `Prod.mk (X := Y) (Y := Y) a`

   (g) `Prod.mk (Y := Y) (fst := a)`

   (h) `Prod.mk b a`

   (i) `Prod.mk (a, b) (b, a)`

   (j) `Prod.mk (snd := Y) (fst := X)`

7. (a) Explain the difference between a `structure` and a `class` in Lean.

   (b) Declarations can have arguments enclosed by parentheses `( )`, braces `{ }` or square brackets `[ ]`. What is the difference between their behavior?

8. In the following questions, your local context is `example {a b c : ℤ} : (a + c) + b = a + (b + c)`. You may find the reference of mathlib lemmas at the beginning helpful.

   (a) Will the tactic `rw [add_comm]` succeed in this state? If so, what does the goal look like afterwards? If not, why not?

   (b) Answer the same question about `rw [add_comm a _]`.

   (c) Answer the same question about `rw [add_assoc a c b]`.

   (d) Answer the same question about `rw [← add_assoc a c]`.

9. In the following questions, assume your local context includes `{k E F : Type*} [NormedField k] [NormedAddCommGroup E] [NormedSpace k E] [NormedAddCommGroup F] [NormedSpace k F]`.

   (a) Explain the error in the following example.

```
example : F → E × F := by
  have := Prod.mk 0 (Y := F)
  exact this
```

   Lean's error is

```
Type mismatch
  this
has type
  F → Prod.{0, u_3} ℕ F
of sort `Type (max u_3 0)` but is expected to have type
  F → Prod.{u_2, u_3} E F
of sort `Type (max u_2 u_3)`
```

(b) In
```
example {s : Set E} : F → E × F := by apply Prod.mk s,
```
explain Lean's error message.

```
Tactic `apply` failed: could not unify the type of `Prod.mk s`
  ?m.3 → Set E × ?m.3
with the goal
  F → E × F
```

(c) In
```
example {a : X} (b : Y) (f : X → Y → X) : Y → X × Y := f a b a,
```
explain Lean's error message
`Function expected at f a b but this term has type X.`

(d) In the following example, explain Lean's error message and discuss how to fix it.

```
example {p q : ℕ → Prop} (h : exists x, p x) (h' : ∀ x, p x ↔ q x) :
    ∃ x, q x := by rw [h'] at h
```

The error message is
`Tactic 'rewrite' failed: Did not find an occurrence of the pattern p ?x in the target expression exists x, p x`

10. For each pair of a true informal statement and a Lean statement below, comment on whether the Lean statement matches the informal statement. If there is a mismatch; explain what causes the mismatch and how you can fix it.

(a) "$5 - 7 + 3 = 1$" with Lean code
```
example : 5 - 7 + 3 = 1
```

(b) "For every integer n we have $(\frac{n}{2})^2 = \frac{n^2}{4}$" with Lean code
```
example (n : ℤ) : (n / 2) ^ 2 = n ^ 2 / 4
```

(c) "If $f : \mathbb{R} \to \mathbb{R}$ has derivative 0 everywhere and $f(0) = 1$ then $f(x) = 1$ for all x." with Lean code
```
example {f : ℝ → ℝ} (h1 : ∀ x, deriv f x = 0) (h2 : f 0 = 1) : f = 1
```