I want to do ______________ in my (cs22) Lean proof but I don't know what tactic to use!

a flowchart of which tactics to use and when to use them

I want to change the way my **goal** looks

You likely want to use an **introduction** rule.
This *constructs the statement* you are trying to prove.

What does your goal look like?

`⊢ A ∧ B` or `⊢ A ↔ B`
and/bi-implication introduction
`split_goal`
will split up your goal into its corresponding subparts

`⊢ A → B`
implication introduction
`assume ha`
will add a hypothesis **`ha`** into your context, remains to show B

`⊢ A ∨ B`
or introduction
`left/right`
will reduce the goal to showing one side of the disjunction

`⊢ ¬A`
negation introduction
`assume ha`
to show that something does not hold, we show that if it did, a contradiction would follow. Likely later in your proof you will use the **`contradiction`** tactic.

`⊢ ∃ e : S, ...`
existential introduction
`existsi x`
will reduce the goal to showing this holds for a specific `x`

`⊢ ∀ e : S, ...`
universal introduction
`fix x`
we fix an arbitrary `x`, and need to show this holds regardless

`⊢ A = A` (equal)
`reflexivity`
will close a goal that is an equals statement that is true by reflexivity

`⊢ A = B` (sets)
`extensionality`
will change the goal to `∀ x, x ∈ A ↔ x ∈ B`, which is the set-element method

other sets related statement
`set_simplify`
will rewrite set operations as logical operations

fact from math
`linarith`
solves equalities and inequalities involving your hypotheses
`polyrith`
solves equalities with polynomials (variables multiplied)
`numbers`
solves facts about numbers
`positivity`
solves something about comparison with 0

`⊢ ∀ n : ℕ, ...`
`basic_induction`
inducts on your goal with standard induction, splits goal into a base case and inductive step
`strong_induction`
inducts on your goal with strong induction, splits goal into a base case and inductive step where your inductive hypothesis is in the form `∀ m < n, P m`

`⊢ ∀ n : ℕ, n ≥ c → ...`
`induction_from_starting_point`
inducts on your goal with standard induction, splits goal into a base case and inductive step, except the base case starts at `c`

I want to change the way my **hypotheses** look

You likely want to use an **elimination** rule.
This *extracts information* from your existing hypotheses.

What do your hypotheses look like?

`hab : A ∧ B`
and elimination
`eliminate hab with ha hb`
will split up your hypothesis into its corresponding subparts

`hab : A ∨ B`
or elimination
`eliminate hab with ha hb`
will split up your hypothesis into two cases, after which two goals will remain, proving when A is true, and proving when B is true

`hab : A ↔ B`
bi-implication elimination
`eliminate hab with hab hba`
will split up your hypothesis into its left and right implications

`hab : A → B`
`ha : A`
implication elimination (modus ponens)
`have hb := hab ha`
given you know A and A → B, will add a hypothesis for B. This is also known as modus ponens.

`hna : ¬A`
`ha : A`
negation elimination
`contradiction`
if you have ¬A and A in your hypotheses, this is a contradiction, so this immediately closes your goal

`hex : ∃ e : S, P e`
existential elimination
`eliminate hex with e he`
will extract a witness out of an extistential statement and a proof that the witness satisfies the proposition

`x : S`
`hall : ∀ a : S, P a`
universal elimination
`have hpx := hall x`
will produce a hypothesis of `P x` from a universal quantification and a specific such value `x` we wish to apply it on. This is going from a general statement to a specific one.

I hope to utilize a definition or equality

`dsimp definition`
will unfold a definition in your goal. For example, **`dsimp dvd`** will apply the definition of "divides".

`rewrite hmn`
if **`hmn: m = n`**, this will replace all **`m`** with **`n`** in your goal. To do the other direction, use **`rewrite ←hmn`**

If you want to use these on a definition on a hypothesis, say **`h`**, you should add **`at h`** at the end of the tactic, like **`rewrite hmn at h`** or **`dsimp definition at h`**

My goal is literally a hypothesis (I'm done!)

`assumption`
will close your goal! check if you have any additional goals that remain to be shown

for complete documentation please refer to **docs.cs22.io**