Sebastian Funk
sebastian.funk@gmail.com

1 Warmup questions

Do these without looking at the notes and in at most three sentences per question:

1. For a strongly-typed programming language, what are the advantages of a type system?
2. Describe the concepts of weakening, exchange and substitution
3. Define the two properties of type safety
4. What is the Curry-Howard Correspondence?
5. Give the typing rules and operational semantics of TLC for functions, pairs and sums

2 Regular questions

Draw proof trees in landscape (if by hand, it helps to start from the bottom). If your proof tree gets too large then break it up into named subtrees. In LaTeX, the `proof` *and* `ebproof` *packages are helpful for drawing the proof trees*

1. Extend the preservation and progress proof to cover AND (exercises 2 and 3 on p38 of the slide deck).
2. Prove type safety for the unit and function cases of TLC (exercises on p66 of the slide deck).
3. In the simply typed λ-calculus, is there any context Γ and type Y for which $\Gamma \vdash x\ x : Y$ (where x is a free variable defined in Γ)? If so, give Γ, Y and show a typing derivation, otherwise prove that no such context and type exists.
4. Under the Curry–Howard correspondence, a proposition P is represented by a type π. Proving P corresponds to providing a term p such that $\cdot \vdash p : \pi$.
   (a) We represent the negation $\neg P$ by a type $\pi \to 0$. Why is this a reasonable representation (using your intuition from propositional logic)?
   (b) Try to prove De Morgan's laws by converting each propositional formula to a type and providing a term of that type in the empty context. *(Hint: not every law holds.)*
      i. $\neg P \wedge \neg Q \supset \neg\ (P \vee Q)$
      ii. $\neg\ (P \vee Q) \supset \neg P \wedge \neg Q$
      iii. $\neg P \vee \neg Q \supset \neg\ (P \wedge Q)$
      iv. $\neg\ (P \wedge Q) \supset \neg P \vee \neg Q$
   (c) De Morgan's laws all hold in propositional logic. What does (b) tell you about the simply typed λ-calculus in relation to propositional logic?

   Note: the simply typed λ-calculus with `abort` *actually corresponds to a form of logic called intuitionistic logic.*
5. Extend logical relation to support products and sums (exercises on p92 of the slide deck).
6. 2023 Paper 8 Question 13
7. 2020 Paper 8 Question 15 part b

3 Extension questions (optional)

1. (a) Find a context Γ under which the term $f\ x\ y$ has type $Bool$, *i.e.* $\Gamma \vdash f\ x\ y : Bool$.
   (b) Can you give a simple description of the set of all such contexts?
2. OCaml has a polymorphic equals-function. That is $equal : a \to a \to bool$ is defined for all types. What are the implications on safety of the type system?