Compilers Honors Exam (2021)

Introduction
You may use any course materials from CS75 and CS73, including all texts, course materials, assignments, lab work, lecture notes, video recordings, and software used or developed as part of the course including both source code and development environments. This test may be taken with pen and paper; however, you are encouraged to use a word processor or text editor for convenience. You may not use other resources including, but not limited to, other software resources or resources on the Internet not previously specified.

Exam
CS75 “Compilers” described techniques to synthesize machine code from a programming language using a strong, dynamic types. CS73 “Programming Languages” explored how adding static types to programming language can allow a compiler to discover type errors before a program is run. This exam asks you to think critically about the impact of type systems on compiler design.

In this exam, “descriptions” and “explanations” should be thorough but may be informal. Prose or pseudocode are acceptable.

Quail
Consider extending Hoopoe with a normalizing static type system without parametric polymorphism or type inference. We will name the resulting language Quail.

1) Define an Ocaml datatype for the Quail AST with type decorations. (Reminder: you may lift entirely from the asts.ml file distributed with Hoopoe and you may use a word processor or text editor during this exam.)
2) Describe your Quail type-checking algorithm.
3) Describe what, if any, optimizations can the Quail compiler accomplish which any Hoopoe compiler cannot? This question is phrased to avoid optimizations which could be added to Hoopoe without static type information.
4) Describe an algorithm that the Quail compiler could use to accomplish these tasks.

Raven
Consider extending the Quail language with parametric polymorphism. We will name this language Raven. As with Quail, Raven does not have type inference.

5) Define an Ocaml datatype for the Raven AST with type decorations.
6) Describe your Raven type-checking algorithm.
7) Explain whether Hoopoe and Raven are operationally equivalent.
8) Describe what, if any, optimizations can the Raven compiler accomplish which any Hoopoe or Quail compiler cannot?
9) Describe an algorithm that the Raven compiler could use to accomplish these tasks.

Flocks
10) Write a paragraph on what kinds of optimizations static types might enable on these user-defined data types:
    - Records
    - Variants
    - Exceptions