CSE P505: Programming Languages
Advice on Approaching Programming in P505
Autumn 2016

You will complete 4–6 (probably 5) homework assignments in P505. Most homework problems will consist entirely of writing OCaml code. OCaml is a great way for expressing relevant concepts because:

- It is more precise and concise than English.
- Its features are well-suited for implementing programming-language "tools" like interpreters, type-checkers, compilers, etc.
- You can make sure your answers type-check, try your answers on example inputs, etc.

Because many of you write industrial-strength code for a living, the act of writing code may obscure the fact that you are completing a homework assignment. **Homework is a very different kind of code:**

- Answers are short and elegant. Many problems will take only a few lines. Less is more.
- Style, aesthetics, and a minimal amount of complexity are the most important things (beyond correctness). The whole point is to clearly and concisely demonstrate understanding of a concept.
- Homework will be related to concepts and examples discussed in class. Obviously the homework will be a little different and have some twists or extensions, but it won't be totally divorced from lecture.
- Quickly throwing small variations at the compiler "until it seems to work" is probably not the best way to understand the material in the course (which is the point of the homework).
- Conversely, going back over a working solution and seeing if there is anything you can do to simplify or cleanse your code is probably a great idea for learning the essence of the material.

Using a mostly functional language like OCaml requires a different mode-of-thinking than programming in C, Java, C#, Perl, etc. This is a good thing — you don't need more practice thinking imperatively, and appreciating functional-programming idioms will make you a better programmer in all languages. So embrace functional programming for your P505 homeworks.

- Variables are not mutable.
- Use recursive functions and library functions like `List.map` instead of loops.
- Use datatypes whenever appropriate.

Relatedly, OCaml is a "real" programming language with lots of features we do *not* need. It will not help you to go learn language features we have not used many times in class. For example, OCaml does have a for loop, but you should not use it. The features you need we will have used multiple times in examples in class, so you can stick to the portions of the language manual that describe these features. **Use the example code from lectures as a guide.**

There may be some *simple* exceptions to this rule. For example, floating-point arithmetic is easy to learn (for example, use `+.` instead of `+`) so we might use it on a homework despite not using it in class. Some library functions for lists might also be useful, but since most of you are new to the language we won't mind you reimplementing library functionality for the most part.

Similarly, you are likely used to industrial-strength IDEs for professional programming, but we really won't need them for the kind of "homework programming" we will do in P505. You are welcome to use any

programming environment you like, but it is probably not an efficient use of your time to tailor a sophisticated environment.

The course staff is available to help with OCaml questions since we certainly do not want you to be stuck on some insignificant detail. We are eager to help and/but asking in certain ways can help us help you. As examples:

- Unhelpful: "Why doesn't this code do what I want?" (We don't know what you want the code to do.)
- Unhelpful: "How can I do such-and-such in Caml?" (We see no evidence of an attempt. Do your best first, and your partial solution will hopefully clarify what you are trying to do, especially if you can explain what you are missing or what isn't behaving as you expected.)

What is helpful is stating in English exactly what you want to compute and how, then including your attempt to translate the English into OCaml.