Homework 2

1. Chapter 1 RQ #1
Q: Why is it useful for a programmer to have some background in language design, even though he or she may never actually design a programming language?
A: The reasons given in the book are:

- Increased capacity to express ideas
- Improved background for choosing appropriate languages
- Increased ability to learn new languages
- Better understanding of the significance of implementation
- Better use of languages that are already known
- Overall advancement of computing

In essence, studying how programming languages work gives the programmer more knowledge, which in turn makes for a better programmer. When all you have is a hammer, everything looks like a nail. By studying language design, programmers are able to expand their toolkit. Sometimes, a hammer is the perfect tool for the job, but it oftentimes isn't. Furthermore, computer science and programming are a relatively new discipline, meaning that change happens all the time. The hammer (i.e the programming language or data structure) you are using today could very well be obsolete by tomorrow. This is less of a problem if (a) you have many other tools to use and (b) understand the fundamentals of language design and thus can pick up the new tools instead.

2. Chapter 1 RQ #16
Q: Why is readability important to writability?
A: Writing requires reading, that is when writing a program, the programmer will have to read the program many times. In addition, a lot of software development today is focused on maintaining existing code rather than writing new code from scratch. If the programming language is highly readable, it will make writing and revising the code much easier.

3. Chapter 1 PS #10
Q: What are the arguments for writing efficient programs even though hardware is relatively inexpensive?

A: There are two main arguments for efficient programs despite cheap hardware. The first is that while hardware can be inexpensive (and continues to become less expensive), it will never be free. Less efficient programs can be orders of magnitude slower and why take write code that is less efficient than necessary? For example, an implementation of a sorting algorithm using bubble sort will take much longer than using quick sort or merge sort for most data sets, and this difference grows larger as the size of the input grows.

The second argument is that while hardware is inexpensive, the programmer's time may not be. Efficient programs will run faster and will therefore take up less of the programmer's and the end user's time. Also, efficient code is more likely to be readable and extensible, which makes it easier for the programmer to maintain the code.