D is for Digital
Summer Reading Guide

This book is a fascinating exploration of all the areas that we will cover in greater detail during the school year. I don't expect you to remember everything or understand everything the first time through. That said, you should find this material interesting and engaging. If you think it's amazing that *everything* we see and do on a computer comes down to sequences of 0s and 1s processed millions of times per second, then you will enjoy this read and the class.

You will often hear me use the phrase "under the hood" next year; this book is the perfect way to begin looking under the hood at a computer and understanding how the technology that we so often take for granted actually works. I hope you truly enjoy this read. As you progress through the chapters, keep in mind the questions below. Mark their answers. Jot notes in the margins. Think of this as an annotation guide. You will not have to turn anything in, but do plan on some sort of assessment the first week of class.

Enjoy this initial dive into the world of computers and computing. I look forward to our journey together in CS50 AP in the fall.[1]

Preface

- What are the three core technical areas of computing? (Hint: they form the three parts of this book.)
- What is a potential fourth, according to Kernighan?

Introduction

- What is one of the many significant technical innovations of the 2000s that Kernighan explains? How did it change and/or improve upon

[1] I am typesetting this document using a language called $\LaTeX$ (pronounced "lay-tech" or "lah-tech"). You will see documents like this throughout our year together. Its overall aesthetics are designed to make documents both more visually appealing and, more importantly, more readable vis-à-vis eye strain. As an example, consider all the room you have for annotations in each margin and the spacing around and within each section of questions.

previous technology?

- Kernighan identifies three important underlying ideas to how computers and communications systems work. What are they?

Part I: Hardware

- Who is often referred to as the world's first programmer? What is she known for?

What's in a Computer?

- What is the role of the CPU?
- If you have a 1.7 GHz computer, what does the "1.7 GHz" part mean?
- What does "mega–" mean? What does "giga–" mean?
- What is the role of RAM? What information does it store?
- What is the difference (or trade-off) between storing something on a disk versus storing something in RAM?
- The most important element of a computer's electronic circuitry is a logic gate. What is a logic gate? (Check out the video for logic gates in the Crash Course Computer Science series on YouTube.)[2]
- What is Moore's Law?

Bits, Bytes, and Representation of Information

- What are the three fundamental ideas about how computers represent information?
- What is the difference between "analog" and "digital"?
- What is a pixel?

[2]DFTBA!

- The iPhone 7 has a 12-megapixel camera and a screen with 1334-by-750-pixel resolution at 326 ppi (pixels per inch). Based on your understanding of a pixel, what do these tech specs mean in simple terms?
- Chances are that you have at least one digital file of music on your computer. See if you can find out its bit rate and sample rate. What are they? (In iTunes, go to "Edit > Get Info > File.")[3]
- What is ASCII? What is Unicode?
- Unicode makes emojis possible. See if you can find the most recently created emojis. What are their Unicode values? (Bonus: how many bits/bytes does it take to represent one Unicode character? How does this contrast to ASCII?)
- What is a "bit"?
- Why would a power switch have a 0 and a 1 on it? What state does each represent?
- If you have N bits, how many different patterns of 0s and 1s can you represent?[4]
- Let's say you have a 1-terabyte hard drive. This could mean two different things (controversially). What are the two possible values for 1-terabyte based on powers of two and powers of ten, respectively?
- What is a "byte"? How many different values can be encoded in 1 byte?
- What is the hexadecimal number system?[5]
- Where is the most common place to come across hexadecimal?

[3]This information is called metadata, a key concept we will address in the course, especially when it comes to music and pictures.

[4]At this point the material might start to seem a little more complex. That's okay for now. Absorb what you can. We will spend a lot of time going over the binary system in class.

[5]This information can be even more complex. Again, understand what you can. We will spend time on it in class. Just be familiar with the concept of hexadecimal numbers.

- Kernighan notes a "critical thing" near the end of section 2.3. Mark this paragraph. It is an idea fundamental to this course. What is his point here?

Inside the CPU

- What is the purpose of an accumulator?
- Kernighan's explanation of the "Toy" computer will be a helpful example when it comes time for us to write and understand code in the C programming language. There are also helpful computational thinking ideas in this section. For example, what is the purpose of "`GOTO`"? "`IFZERO`"?
- The bottom of page 39 has some essential ideas for how to construct programs. What key ideas emerge here?[6]
- In very simple terms, what is the fetch-decode-execute cycle?
- In the context of computer memory, what purpose do caches serve?
- Kernighan gives you a fun Google trick to try at the end of 3.3. Come up with an obscure, seemingly random phrase, and note the time it takes Google to respond. You'll see this at the top of the results. Then, run it again immediately. What is the time differential? Caching at work![7]
- Who was Alan Turing?[8] What is the "Turing test"?
- What does CAPTCHA stand for?

---

[6]Notice how you don't need to program to get better at the logic of programming; the key to being a good programmer is not knowledge of syntax but logical and ordered thinking.

[7]I Googled the phrase "hippopotamus homeric coffee." The first search took 1.03 seconds, the second only 0.41. See how high you can get the response time.

[8]You may also know him as the character played by the inimitable Benedict Cumberbatch in *The Imitation Game*.

Part II: Software

- What is software? How is it different from hardware?

Algorithms

- How does Kernighan define "algorithm"?
- Kernighan gives a brief aside to data structures. What is his short explanation for this term?
- What does the term "linear-time" have to do with algorithms?
- What is binary search? What real world application does it have?
- Sorting algorithms will be a focus of our study later in the first semester. For now, which algorithm is better? In simple terms, how does this better algorithm work? That is, what is its general approach to sorting?
- What is the Traveling Salesman Problem?
- Look up the word "heuristic" in the context of computer science. How does it relate to the end of 4.4?

Programming and Programming Languages

- What is the difference between an algorithm and a program?
- What is the function of an assembler and assembly language?
- Are assembly language instructions the same for all computer processors?
- What is the function of a compiler?
- What are the advantages of high-level languages relative to assembly language?
- What are the five high-level languages Kernighan demonstrates code with? Which seems hardest to read and understand? Easiest?[9]

[9] For your reference, CS50 uses C and Python as its primary languages. Other than a quick introduction to programming using Scratch, all of first semester will be spent focused on learning C.

- What is a library? What is a library's API?
- What is the term for a flaw in one's code? What is the origin of this term as it relates to computers?
- What are general definitions for the following terms: trade secrets, copyright, patents, licenses?
- What is a "standard" in the world of computing?
- What is the relationship between source code and object code?
- What does it mean for code to be "open source"?
- What is one example of open source software? (Hint: I'm typing this on a Chromebook, which relies significantly on open source.)

That's all I want you to read: 5 chapters. Really. Stop at page 83 (unless you simply can't put it down in which case read as much as you like). That's about the first 1/3 of the book. We will use the latter 7 chapters throughout the school year. Because of its relatively short length as summer reading assignments go, don't read this just once. It is certainly dense and introduces many concepts which may be brand new to you. I encourage lots of frequent Googling. Dive down some Wikipedia rabbit holes. Explore the tech specs of your own computer(s). (For example, what is the size of your L1 cache?) Read about the history of Linux (which our CS50 IDE[10] uses as its operating system). These 5 chapters will prepare us to jump right into Unit 0 on Day 0 in August.[11]

As I said above, I cannot wait to start CS50 AP with all of you. Enjoy your summer. Learn a little in between long bouts of sleep and Netflix. See you in the fall!

[10]IDE stands for Integrated Development Environment. In simple terms it is where we write and execute our programs.

[11]Computer scientists start counting at 0, not 1.