Computer Science 50
Fall 2010
Scribe Notes

Week 0 Friday: September 3, 2010
Andrew Sellergren

Contents

1 Announcements (0:00–10:00) 2

2 Excerpts from the Syllabus (10:00–15:00) 2

3 Shuttleboy (15:00–18:00) 3

4 From Last Time (18:00–20:00) 3

5 Binary and Hard Drives (20:00–45:00) 3

6 Computer Programs (45:00–75:00) 6

6.1 Our First in Pseudocode: The Socks Problem 6

6.2 Our First in C: Hello World 7

6.3 Our First Few in Scratch 8

1 Announcements (0:00–10:00)

- This is CS50.
- 0 new handouts. 3 old handouts. 1 floppy disk.
- Typically, we won't have class on Fridays except for this week and next week.
- By the end of this course, you'll be able to "create a GUI interface using Visual Basic and track an IP address." Or, actually, you'll be able to understand why this clip from CSI is utterly absurd. While we're making fun of CSI, take a look at the infinite zoom capabilities of their photo software!
- Finally, a non-spoof. Check out Gmail Priority Inbox! From a computer science perspective, this is an interesting problem. Gmail can "learn" what e-mails are important to you based on many different heuristics, e.g. which e-mails you actually read, which e-mails you reply to, which e-mails are addressed directly to you.
- Videos as well as full, searchable, clickable transcripts of lectures will be available within a few days of the lectures having been given.

2 Excerpts from the Syllabus (10:00–15:00)

- "what ultimately matters in this course is not so much where you end up relative to your classmates but where you, in Week 12, end up relative to yourself in Week 0"
  - That means that when calculating grades at the end of the semester, we consider your prior experience as well as your upward trending.
- "can take CS50 Pass/Fail or for a letter grade"
- "the course is not graded on a curve"
- "those less comfortable and somewhere in between are not as a disadvantage vis-à-vis those more comfortable"
- "each student's final grade is individually determined at term's end after input from the teaching fellows"
- psets
  - 9 late days
  - drop lowest score
  - standard + Hacker editions
- support structure

  - scribe notes[1]
  - searchable transcripts
  - 150+ office hours per week
  - support forums
  - help@cs50.net

3 Shuttleboy (15:00–18:00)

- Check out Shuttleboy, an interactive, web-based, SMS-based, voice-based applcation for determining when the next shuttle is
- Dial 617-BUG-CS50 and follow the voice prompts! (actually, don't, because only one of them is currently active, but the rest are available as potential final projects!)
- Sometimes technology isn't the answer. To that end, there are also Shuttleboy cards that display the entire shuttle schedule on a laminated card that fits in your wallet. Preorder yours here.

4 From Last Time (18:00–20:00)

- The Unofficial Guide to Computer Science at Harvard is designed to help you navigate your way to a CS major or minor.
- HarvardFML is another CS50 final project that we like showcasing. Here's a quote about CS50 taken from it:
  - "I talk about cs50 so much that my teammates implemented a cs50-talk tax. I get charged $1 for each mention of the class, I already owe $20 dollars and haven't been able to talk for more than five minutes without getting called out on it. FML."

5 Binary and Hard Drives (20:00–45:00)

- You're familiar, of course, with the decimal system, which represents a number like one hundred twenty three as "123," with a one in the hundreds column, a two in the tens column, and a three in the ones column.
- Binary is very similar except that the columns represent powers of two rather than powers of ten. Thus, the rightmost column is the ones column, the second-to-rightmost is the twos columns, the third to rightmost is the fours column, and so on.
- In binary, the only digits that are available to us are 0 and 1, as compared to decimal in which the numbers 0 through 9 were available.

[1]Snarky comment!

- So how might we represent the number 7 in binary? 0111. Note that leading zeroes are inconsequential.
- The fact that 0 in binary is still written as 0 is the fundamental reason why computer scientists begin counting from 0 rather than 1. Otherwise, it would waste a bit (short for binary digit).
- Lest you think that binary is merely a pedagogical exercise, realize that we'll be revisiting it in order to solve the practical matter of how to recover JPEGs from a formatted flash drive. Those JPEGs are stored on disk as a series of zeroes and ones.
- Now that you understand binary, you'll get the joke behind the Foxtrot comic in which Jason professes to have completed his phys-ed homework, which required him to do 100 pushups, having done only 4.
- In order to use binary to express alphabetic characters, we'll need some kind of mapping between numbers and letters. This is where ASCII comes in. To see the complete mapping, check out this ASCII table. However, you only really need to know that uppercase A is 65 and lowercase a is 97. This will help with conversions.
- If we allow ourselves eight columns, we can represent all the numbers from 0 to 255 (and their character mappings according to ASCII) in binary. These eight columns correspond to eight bits, which together make up a single *byte*. The letter A in binary is written as 01000001.
- Why use binary rather than decimal as the language for computers? Consider that in binary, we have only two digits that we need to implement. We can call the digit 1 "on" and the digit 0 "off." In this way, we can sync our counting system with electricity.
- To hammer this point home, we'll bring eight volunteers on stage, one each to represent the digits of a byte. When a volunteer raises his hand, his digit is "on." After three rounds of this, we can spell out the word B-O-W.
- Wooly Willy is a children's toy that allows you to decorate a man's face using a wand and magnetic particles.[2] What's relevant about this toy is that your computer's hard drive similarly has a large number of magnetic particles on its circular platters. When a magnetic particle is aligned in one direction (say, with its North end pointing upward), it represents 1. When it's aligned in the opposite direction, it represents 0.
- Check out this video and this follow-up which explain more technically how hard drives work. Here are the take-away points of these videos:

[2]Note that because David is 60 years old, you may never have heard of this toy. You probably played video games instead.

  - Hard drives consist of circular platters which spin as data is being written to and read from them. When data is to be stored on the hard drive, it passes from RAM along with software signals that designate how the data is to be stored. Certain signals control how the platters spin and others control the read-write heads. The distance between the heads and platters is less than the width of a human hair, yet the platters spin 5400 RPM or faster. Multiple platters make for greater efficiency than a single platter.
  - A read-write head in a hard drive contains a tiny electromagnet which conducts the software signals it is passed and, during writing, flips single bits on the platter either on or off by polarizing it in one direction or the opposite. During reading, the process is reversed and the electromagnet conducts signals from the bits on the platter. A single file may be stored in locations scattered widely across a platter, so a separate file keeps track of the locations of all the bits of files.

- To see something like a hard drive platter firsthand, take the floppy disk we handed you at the beginning of class and pry off the metal shutter. Now break apart the plastic covers and reveal a thin circular disc. This is where all the data is stored! Go ahead and corrupt some bytes by running your fingers around on it.

- Although your computer's hard drive is roughly the same size as that floppy disk, it can probably store a million times the data. A floppy disk holds 1.4 megabytes (1.4 million bytes) whereas many hard drives store more than 1 terabyte (1 trillion bytes).

- Question: what happens when you format a hard drive? Not much, actually. Formatting only modifies a few bytes on the hard disk, namely those that record the locations of files. The files themselves are left intact, which is why forensic data recovery is possible. This is precisely what you will do this year when you recover a set of JPEGs from David's formatted flash drive![3]

- Question: what does it mean to defragment a hard drive? When saving files to your hard drive, they might be written to consecutive bytes on disk. However, if you delete any files in that series, there will be some bytes that are left orphaned. Modern operating systems allow you to reclaim those orphaned bytes and consolidate your remaining hard drive space. The idea behind this is to speed up access to your files, since fragmented files, or files whose data is scattered across multiple disk locations, are slower to open. Defragmenting is less necessary these days because modern hard drives are much more high-performing and can access even fragmented files very quickly.

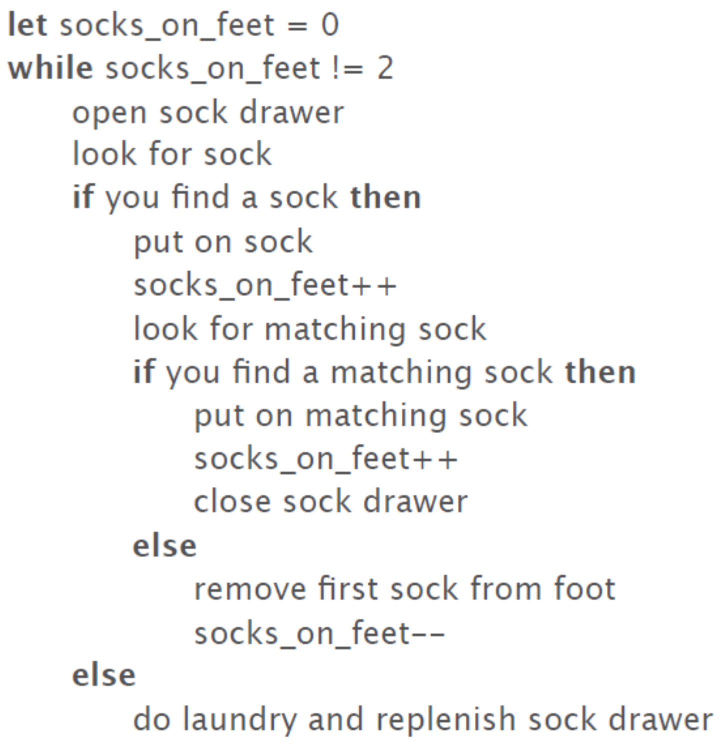
[3]You'd think someone with a PhD would be smart enough not to accidentally format his flash drive year after year. You'd think that, but you'd be wrong.

- Question: how is the read-write head of the hard drive controlled? A small circuit board along with software drivers are how the operating system instructs the read-write head to physically move in different directions across the platters.
- Question: what are the advantages of SSDs (solid-state drives)? SSDs have no moving parts and are akin to large flash drives. Read-write access to these disks is much faster than typical hard drives.

6 Computer Programs (45:00–75:00)

6.1 Our First in Pseudocode: The Socks Problem

- Let's say we want to write a computer program for putting our socks on in the morning. We'll write this program in pseudocode, a not-quite programming language that allows us to express human-readable instructions that can easily be translated into real code. Our socks program looks like so:



```
let socks_on_feet = 0
while socks_on_feet != 2
    open sock drawer
    look for sock
    if you find a sock then
        put on sock
        socks_on_feet++
        look for matching sock
        if you find a matching sock then
            put on matching sock
            socks_on_feet++
            close sock drawer
        else
            remove first sock from foot
            socks_on_feet--
    else
        do laundry and replenish sock drawer
```

- In step 1, we set a variable `socks_on_feet` to the number 0.

- In step 2, we enter a loop that will be repeated as long as the condition (`socks_on_feet` does not equal 2) is true.
- Inside the loop, we open the sock drawer, look for a sock, and then enter a condition:
  - If a sock is found, put it on.
  - Otherwise, do laundry and replenish sock drawer.
- Note that our code is indented in key places. Everything which is encapsulated in the `if` condition is indented to indicate so. The computer most likely doesn't care about this indentation but it makes the code easier for us to read.
- The `++` and `--` notation are shorthand for "add one to (increment) this variable" and "subtract one from (decrement) this variable" respectively.
- This code has a bug. If only one sock is in the sock drawer (or none of the socks has a matching partner), it will remain in the while loop forever.
- This is called an infinite loop. As you will see later, sometimes programs we write will contain infinite loops. We will recognize these cases because when we run the code, the computer will "hang"–it will keep running and produce no output because it is stuck in a loop.
- It might seem obvious to us what to do in the case where there is only one sock in the drawer, but a computer cannot make assumptions the way we can.
- Therefore, our algorithms must be extremely specific and precise and account for all possible cases. Nothing can be assumed.

6.2 Our First in C: Hello World

- Let's open a blank text file named `hello.c` and write our first program in C:

  ```
  #include <stdio.h>

  int
  main()
  {
      printf("hello, world\n");
  }
  ```

  Forgive the cryptic syntax for now, note that you can probably guess what the effect of this program is: to print the text "hello world."

- The `include` line tells our program to borrow some code from someone else's library. This way, we don't have to reinvent the wheel every time we want to print to the screen, for example.

- The next step is to pass this file to the compiler. The compiler is responsible for translating our human-readable code into machine-readable instructions. Whereas our code is in plain English (sort of), the compiler's output will be in binary.

- Once we run the compiler, a file named `a.out` will be created. This is our program! When we run it, we see that "hello, world" is printed on our screen.

6.3 Our First Few in Scratch

- To start working with Scratch, you'll need to download the program from MIT's website.

- Once you install Scratch and open it, take note of the following layout:

  - On far left, notice the "palette" of puzzle pieces, which represent programming statements. Programs will be composed by putting puzzle pieces together in a particular order.
  - At bottom right are sprites, or characters that will carry out your instructions.
  - At top right is the stage, where the program will be carried out.
  - To the left of the stage is the scripts area, where puzzle pieces must be dragged and strung together.

- We can recreate our very simple C program in Scratch using the "say" block. `Hai1.sb` is equivalent to `hello.c`, only a little more colorful. The "when green flag clicked" piece is equivalent to our `main` keyword in the C program above.

- Before we go any farther, let's talk about some computer science jargon: a *statement* is an action that we give to the computer to perform. In the context of Scratch, statements begin with verbs like "say," "play," and "wait."

- Obviously, we're taking baby steps, but realize that that's what programming is all about–taking very basic building blocks and creating functions and more complicated programs.

- `Hai2.sb` is slightly more complicated. The cat will say "O hai, world!" for 1 second, wait 1 second, say it again for 1 second, wait 1 second, and say it again for 1 second.

- So far we've only made use of *statements*, which are direct imperatives given to the computer. But if we want to introduce logic into our program, we'll need *boolean expressions* and conditions. Boolean expressions are those that have only two possible values: true or false, yes or no, on or off, 1 or 0. No matter how you say it, it's a simple variable. In Scratch, boolean expressions are represented as hexagons and are written as yes-or-no questions such as "touching mouse-pointer?," "mouse down?" or comparisons such as less-than, equal-to, or greater-than.

- Boolean expressions aren't new to you. Consider on HarvardCourses when you search for courses on Mondays and Tuesdays in the fall. Somewhere in the code, that translates to Monday `and` Tuesday `and` fall.

- We can take different courses of actions in different conditions. In a certain condition, when a variable is less than 5, we take some action. In another condition, when that variable is greater than 5, we take a different action. To do this, we use if-then statements. You can also nest these conditions so you can take more than two courses of actions.

- `Hai4.sb` and `Hai5.sb` make use of conditions and boolean expressions. In the first, the condition `1<2` always evaluates to true, so the cat meows everytime we click the green flag. In the second, however, the condition says, "pick a random number between 1 and 10 and if that number is less than 6, have the cat meow." This is what we call a pseudorandom number generator. Although it seems simple here, the idea of forcing a deterministic machine to non-deterministically generate a random number is actually quite complicated. There is an entire branch of study devoted to this very task. One theory holds that random numbers can be generated from the white noise picked up by a microphone. In any case, the effect of this pseudorandom number generator on our program is that the cat will meow approximately half the time we click the green flag.

- If we want our cat to meow multiple times, we can certainly just duplicate the statements however many times we want. But this has several disadvantages:

  - It is resource-inefficient.
  - It is tedious.
  - It makes it difficult to change what the cat is saying.

  Remember our goal is not just to accomplish a task, but to accomplish it elegantly and efficiently.

- To that end, we can use loops when we wish to repeat a statement. In `Hai6.sb`, we implement a loop which causes the cat to meow indefinitely. Infinite loops aren't necessarily bad: consider the case of a spellchecker in a word-processing program that constantly checks for words you have typed.

- In `Hai7.sb`, we combine a loop and a condition so that the cat will meow only if we the mouse pointer is touching it or, in other words, if we are petting it. In `Hai8.sb`, we add an extra condition so that the cat will meow indefinitely, but will roar if we touch it with the mouse pointer.

- *Variables* are another useful programming construct. They allow us to store information about the *state* of a program. For example, in the case of the socks algorithm, the variable `socks_on_feet` stored the number of socks on our volunteer's feet.

- *Arrays* are essentially collections of related variables. In the game `FruitcraftRPG.sb`, for example, an array is used to store the different types of fruit which have been collected.

- Now is a good time to introduce the concept of *threads*. This is a fairly complicated concept which doesn't usually get introduced in the first week of a computer science course, let alone in the first semester of computer science training. However, let's Scratch the surface of a threading discussion.[4]

- Threading refers to the notion of multiple threads of code executing simultaneously. In `Move1.sb`, we have a simple animation of a duck moving back and forth across the screen, screaming and turning around every time it touches the edge. This is a single thread. In `Move2.sb`, we achieve "multithreading," at least in appearance. In terms of programming, we have two different scripts associated with two different sprites, a cat and a bird. For the cat, we begin by placing him in a given spot on the stage and orienting him in a random direction. Then we begin a loop whereby if he touches the bird, then the game ends; otherwise, orient toward the bird and advance one step. For the bird, we again place and orient him and then move him around the stage three steps at a time. Effectively, then, the cat is chasing the bird until he catches him.

- *Events* are another method of communicating between sprites. `Marco.sb` leverages events to play the game of Marco Polo.

- Scratch also offers sensor boards which take user input in the form of sound, light, and movement, as demonstrated by `singer.sb`, `Masquerade.sb`, and `davidwu.sb`, respectively.

- Check out `Oscartime.sb` for another example of what you can do with Scratch!

- Good luck with Problem Set 0! Check out the course website for a schedule of office hours if you're having trouble!

[4]Yes, yes I did just make that joke.