1. Announcements
2. Some review
   a. State machines
      i. Controls on your iphone
         1. Assume 3 buttons
            a. One turns screen off or on
            b. Others turn volume up or down
         2. So, six states
            a. Screen on or off
            b. Volume at low, medium or high
         3. Buttons cause transitions
   b. Logic problem that use our operators
      i. Various constraints
         1. Frank will go to the party if Ed goes AND Dan does NOT.
         2. Dan will go if Bob does NOT go OR if Carole goes.
         3. Ed will go to the party if Alice AND Bob go.
      ii. Setting some values
         1. Alice and Bob decide to go, but Carol stays home.
      iii. Questions
         1. Will Frank go to the party?
      iv. Equations
         1. F = E AND NOT D
         2. D - NOT B OR C
         3. E = A AND B
      v. Filling in values
         1. A = B = 1 ; C = 0
         2. E = 1
         3. D = 0
         4. F = 1
3. One last class of algorithms
   a. To add to logarithmic, linear, almost linear, quadratic
   b. Exponential
      i. If input size grows by 1, work needed doubles
      ii. Example problem, Towers of Hanoi
         1. 3 stacks
            a. At stack, a collection of tubes on first stack each larger than the one above
            b. Have to move them to the 3rd stack
               i. Can never put larger on top of smaller
               c. Can use 2nd stack as intermediate storage
         iii. Simple case
            1. 3 tubes A < B < C
            2. Move A to 3rd, B to middle, A to middle, C to 3rd, A to first, B to 3rd, A to 3rd
3. If 4 tubes, have to move them to get top 3 in order on middle (7 moves), then move 4th (D) to 3rd post, then have to move top 3 to 3rd post (7 more moves) for total of 15 moves

4. If 5 tubes, 15 to move top 4, 1 to move 5th, 15 to move top 4

5. ... If n tubes, $2^{n-1}$ moves are needed.

4. There is another class of problems that are of interest
   a. Not known to be exponential but not known to be easier
   b. If any one of the class of problems is easy, then all are
   c. Problem is known as the P vs NP question (or P=NP?)
      i. If you can either find a fast method or prove that none exists, you will win $1M
      ii. Side of computer science building has bricks turned
d. Sample problem
   i. Knapsack problem
      1. Given a collection of weights and a knapsack of fixed capacity, can you exactly pack the knapsack?
   ii. Traveling salesman problem
      1. If a salesman wants to take a tour of various cities with certain distances between pairs, what is the shortest distance that the salesman can travel.

5. Review of algorithms
   a. Different running times of algorithms
      i. Logarithmic
         1. Double problem size, add 1 operation
      ii. Linear
         1. Double problem size, double number of operations
      iii. Quadratic
         1. Double problem size, quadruple number of operations
      iv. Exponential
         1. Add 1 to problem size, double number of operations
      v. Almost linear (N log N)
         1. Double problem size, number of operations slightly more than doubles
   b. Algorithms for various tasks
      i. Searching
      ii. Sorting
      iii. Finding largest, jokers, ...
      iv. Checking credit card number to see if it is valid
      v. More complicated processes
         1. Packing a knapsack
         2. Towers of Hanoi
   c. Defining algorithms -- Algorithm must
      i. Be precise and unambiguous
      ii. Cover all possible situations
      iii. Never get to a situation where it doesn’t know what to do next
      iv. Eventually stop and not run forever (in most situations)
6. Creepy or not creepy
   a. https://arxiv.org/abs/1510.03346 discusses ethical dilemma in designing an autonomous vehicle (self driving car)
      i. If choice between hitting pedestrians in crosswalk or hitting tree and damaging car and driver
         1. People say that the least damage to people should be done
         2. People say that their car should be set to protect themselves
      i. Give cars a button that ranges between full altruist and full egoist

7. Programming
   a. Algorithms give recipes
      i. Precisely stated
      ii. Unambiguous
      iii. Defined sequence of steps and conditions for stopping
      iv. idealized
   b. Programs give their realization
      i. Convert algorithm into a form the computer can process
      ii. Difference between blueprint and building
         1. computer may cause small changes to be required
            a. finite (or inefficient) memory
            b. limited speed
            c. erroneous data
            d. different interpretation of language on different machines
      iii. A program is meant to be a guaranteed recipe for a cooking robot
   c. Programming languages have certain characteristics
      i. Ability to assign values to variables and store/load them
         1. LOAD MEM and STORE MEM in Toy
      ii. Ability to do arithmetic operations
         1. +,-,*,/ maybe also **, log, sine, ...
      iii. Constructs that allow for looping
         1. IFZERO, IFPOS, GOTO in Toy
         2. Often
            a. WHILE (condition is true) DO ....
            b. IF (condition is true) THEN ... ELSE ....
            c. FOR (values of a variable) DO ...
      iv. Potentially sophisticated ways to structure data
      v. May also have some builtin operations of more (or less) sophistication (APIs)
   d. What does a programming language do?
      i. A compiler translates (in steps) the program from a higher level language into assembly language
         1. First, parse every statement to make sure it has legal structure within the language description
1. Is a characteristic of higher level languages at the cost of some efficiency
2. Makes programs portable

8. How have programming languages evolved

a. 1940’s -- Machine level
   i. Programming at the level of the machine
   ii. Writing code in binary
   iii. Harder to write correct programs
   iv. Machines were slow and higher level languages didn’t exist

b. 1950’s -- assembly language
   i. Similar to the Toy language
   ii. Written with mnemonics for instructions
      1. Better than machine language but still lower level
      2. Could say ADD instead of 0110101, ...
   iii. Translated into machine language for execution

c. Late 1950’s – first higher level languages
   i. Fortran (Formula Translation) created at IBM by John Backus
   ii. Cobol (Commercial Business Oriented Language) sponsored by DoD created by Grace Murray Hopper
   iii. Written for special purposes
      1. Fortran for mathematical work
      2. Cobol for business work
   iv. Both have been refined over time and still exist
   v. Advantages
      1. Writing in something much closer to English
         a. A = B + C instead of GET; STORE B; GET; ADD B; STORE A; PRINT
2. Much easier to move programs from one machine to another

vi. Disadvantages
   1. Less efficient in use of machine

d. Moving into the 60’s
   i. Algol (Algorithmic Language)
      1. Developed by an international committee
      2. Designed to be the universal language
      3. Sabotaged by IBM
   ii. BASIC (Beginner’s All-purpose Symbolic Instruction Code)
      1. Developed at Dartmouth in 1964 by Kemeny and Kurtz
      2. Designed to be so simple that anyone could program
   iii. LISP
      1. Designed by John McCarthy as a practical notation for computer programs
      2. Evolved into the language of choice for AI for some decades

e. Moving to the 70’s
   i. C
      1. A language that allowed for (low-level) systems programming in a higher level language
      2. Original versions of UNIX were written in C
      3. Led to many later languages – C++, C#

f. The 80’s – object oriented programming
   i. C++
      1. Object oriented approach was designed to make larger programs easier to write
         a. Structures for holding data are grouped with functions that operate on the data
      2. Led to many later languages (e.g. Java)

g. The 90’s – scripting languages
   i. Write big programs by combining pieces already written
   ii. Often written on virtual machine that is simulated (like Toy)
   iii. Aimed to be more secure
   iv. Easier to program in; more portable; slower (but machines were faster)
   v. Small snippet of javascript
      1. var sum = 0
      2. var num = prompt(“enter new value or 0 to end”)
      3. while (num != 0) {
         a. sum = sum + parseInt(num)
         b. num = prompt(“enter new value or 0 to end”)
         c. }
      4. alert(“Sum = “ + sum)

h. moving into the 21st century
   i. new general purpose languages
      1. Go, Rust, Swift, Scala
2. Old languages never die
   a. New refinements to C, C++, Fortran, Cobol, ...
3. Specialized languages for specific applications
   a. E.g. R for statistics
 ii. Ultimately natural language