1. Announcements
   a. Problem set 3 and lab 3 are available
   b. Problem set 2 due today
      i. How was/is it?
   c. Lab 2 due Friday
      i. Common challenges
         1. Smart quotes
         2. Getting things onto cpanel
            a. Difference between download and seeing the website
               i. Netid.mycpanel.princeton.edu and has to have files uploaded
         3. Seeing your html in TextEdit
2. Moving into the computer and making it do things
   a. Architecture of the CPU
      i. Contains Arithmetic Logical Unit (ALU) to do the work
      ii. Contains an accumulator (ultimately registers) where work is done
      iii. May have some memory (cache memory)
      iv. Connects to main memory (RAM) via LOAD and STORE
      v. Connects to keyboard/screen via GET and PRINT
   b. Architecture of the RAM
      i. Just a bunch of memory locations that can be individually addressed
   c. CPU will do the work; RAM will store the information
      i. CPU does 2 things
         1. Fetch
            a. Next instruction
            b. Data for current instruction
         2. Execute
            a. Computation (usually simple) for current instruction
         3. A finite state machine controls the work
            a. CPU
               i. Is it fetching or executing
               ii. Where does it fetch next
            b. ALU
               i. What operation is it doing
                  1. Takes operands
                  2. Writes result
         4. An accumulator holds the data being considered
            a. Can LOAD into the accumulator from RAM
            b. Can SAVE accumulator into RAM
            c. Can GET information from the keyboard
            d. Can PRINT value in accumulator to the screen
            e. Can ADD the value from a location in RAM to the value in the accumulator and store it there.
      ii. RAM stores 2 things
1. **Instructions**
   a. Simplest instructions are LOAD, SAVE, GET, PRINT, ADD, STOP

2. **Data**
   a. Variables can be named and stored, loaded by name

3. **Key architectural insight – no distinction between instructions and data**
   a. Pluses – makes the architecture much more flexible
   b. Minuses – no boundary; can read data as an instruction and it may not make sense
   c. Known as von Neumann architecture after John von Neumann
      i. Invented in Princeton at IAS
   iii. For the moment imagine that RAM stores either words (our basic instructions) or data (values)

   1. First (and simplest) program
      a. GET
         i. Opens a box that asks for a value; puts that value into the accumulator
      b. PRINT
         i. Writes the value of the accumulator to the screen
      c. STOP
         i. ends

   2. What if we want to read 2 values and add them
      a. Accumulator can hold only one number
         i. This program doesn’t work
            1. GET
            2. GET
            3. ADD
            4. PRINT
            5. STOP
         ii. Second GET overwrites the value in the accumulator
      b. Have to define variables to store values
      c. Program
         i. GET
         ii. STORE NUM
         iii. GET
         iv. ADD NUM
         v. PRINT
         vi. STOP
         vii. NUM
         viii. Important note – instructions cannot be in very first column; labels (in this case NUM) must be there

3. **Creepy or not creepy**

4. **Quantitative moment**
How many trees on the Princeton campus?

5. Back to Programming
   a. Adding 3 numbers
      i. GET
      ii. STORE NUM1
      iii. GET
      iv. STORE NUM2
      v. GET
      vi. ADD NUM1
      vii. ADD NUM2
      viii. PRINT
      ix. STOP
      x. NUM1
      xi. NUM2
   xii. Eventually this gets tiring
   b. If you want to add a bunch of numbers, it gets complicated
      i. Especially if you don’t know how many or when to end
         1. Examples
            a. Add numbers until you see the number 0
            b. Read a number and add that many numbers
      ii. We’re going to need some ability to do tests and act on the results
   c. Introduce flowcharting
      i. Programs can be straightline
         1. Do this then this then this ...
      ii. Or have looping
         1. So then then this and then test;
            a. Based on the test either do this or that
         2. Flow may not be straightline; may jump to a different place
   d. Back to adding numbers until you see 0
      i. Create a flow chat
         1. Boxes that move you forward (in straightline fashion)
         2. Boxes that are decision boxes
         3. Arrows that direct the flow from statement to statement
      ii. To program this will need new constructs
         1. Programming language will have to have a facility for putting looping in
         2. Need to add instructions
            a. IFZERO XXX – tests to see if the value in the accumulator is zero.
               i. If so, go to the instruction labelled XXX
               ii. If not, go to the next instruction
            b. GOTO XXX
               i. Instead of going to the next instruction, go to the
                  instruction labeled XXX
   e. Review examples and entire instruction set
      i. GET; PRINT
ii. LOAD Val; STORE M
iii. ADD Val; SUB Val
iv. GOTO L; IFPOS L; IFZERO L
v. STOP
vi. M NUM

f. Now look at program to read a number N and add the next N numbers that are input

6. We've programmed now
   a. How are programs processed?
   b. What is the difference between what we are doing and real programming?
      i. We had only 1 accumulator
      ii. We had to manage memory
      iii. Our programs were, of necessity short

7. Give insights into what it is that they will have to do to write their programs