Final Review
• Lecture 1
  – Seven Big Ideas in computing
  – Computational Principles, Computational Thinking
  – Historical and current gender trends in CS
  – Exponential increase in computing power
  – Big Data
Aim: Cover the Seven Big Ideas in computing

1. Computing is a **creative human activity** that enables innovation
2. **Abstraction** is a way to understand and solve problems
3. **Data and information** help to create knowledge
4. **Algorithms** are tools for developing and expressing solutions to computational problems
5. **Programming** is a creative process that produces computational artifacts
6. **Digital devices**, systems, and the **networks** that interconnect them enable and foster computational approaches to solving problems
7. Computing enables innovation in **other fields** like sciences, engineering, humanities, etc.

– We want your feedback about what in computing you find exciting and interesting
Two Aspects to the 7 Big Ideas

- **Computational Principles**
  - 'Bits can represent all information'

- **Computational Thinking**
  - Thinking approaches you can use to solve your problems with computers
Why teach a new version of CS Principles?

Figure 1. Computer Science Listed as Probable Major Among Incoming Freshmen
Source: HERI at UCLA
Computing as part of our lives

- Computing capacity
  - Increasing 58% annually
  - Doubling every 18 months
- Telecommunications capacity
  - Increasing 28% annually
  - Doubling every 34 months
- Storage capacity
  - Increasing 23% annually
  - Doubling every 40 months
- Expanding at these rates for decades
The full scale of how much information we make is hard to appreciate. We humans collectively now have capacity to store approximately 300 exabytes of information. This is close the total amount of information stored in one person’s DNA. Or, as Hilbert puts it, it’s the equivalent of 80 Libraries of Alexandria per person on the planet. And remember, the technium is doubling its capacity every year and a half, and your DNA is not. Broadcasting has grown at about the same speed as world’s GDP; but our information storage capacity has grown 4 times faster and telecommunication capacity has grown roughly 5 times faster than the world’s economic power.
Lecture 2

- LightBot – agent
- Abstraction
- Recursion
- Computing mechanically
- McCulloch & Pitts neuron
- Truth tables
Other aspects of “commanding”

- The **agent** is usually a computer, but can be a person or device (like a robot)
- The agent follows the commands, the **instructions**, **without making mistakes**, **without stopping**, doing only what is asked
- The program implements your intentions – it does what **you tell it to do**
- **You** are trying to get the robot to the blue tiles and flip them to yellow
The word “abstraction” is used a lot in computer science.

It's one of the Big 7 Ideas.

Abstraction is a way to understand and solve problems.

Abstraction: eliminates details to focus on essential properties.

The instruction example demonstrates functional abstraction – that means we have given a name to a series of operations that perform a meaningful activity.
Why doesn't it go on forever?

drawTris(u, depth, new PVector(dorder, n-border), new PVector(width/2, border), new PVector(width-border, n-border));

void drawTris(int level, int maxLevels, PVector left, PVector top, PVector right)
{
  level++;
  if (level >= maxLevels) {
    triangle(left.x, left.y, top.x, top.y, right.x, right.y);
  }
  else {
    PVector a = PVector.add(left, PVector.div(PVector.sub(top, left), 2));
    PVector b = PVector.add(right, PVector.div(PVector.sub(top, right), 2));
    PVector c = PVector.add(left, PVector.div(PVector.sub(right, left), 2));
    drawTris(level, maxLevels, a, top, b);
    drawTris(level, maxLevels, left, a, c);
    drawTris(level, maxLevels, c, b, right);
  }
}

void mousePressed()
{
  switch(mouseButton) {
    case LEFT: depth++; break;
    case RIGHT: depth = max(depth-1, 0); break;
  }
  redraw();
}
When the brush touches the roller, closes the circuit

Electrical impulse can cause a mechanical action to happen
**McCulloch & Pitts Neuron**

Mathematical formula:

\[
\text{Input} = X_1 \times W_1 + X_2 \times W_2 + \ldots + X_n \times W_n
\]

Output:

\[
\text{Output} = 1 \text{ if Input} > \text{Threshold}, \ 0 \text{ otherwise}
\]
<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>P and Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>
Lecture 3

- ASCII
- Privacy, Principles of Fair Information Practice
- Digital footprint
- Terms & Conditions, privacy settings
- Snowden
<table>
<thead>
<tr>
<th>ASCII</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

---

0100 0011  C
0100 0001  A
0101 0100  T
0100 1100
0100 1111
0100 1100

---

<table>
<thead>
<tr>
<th>ASCII</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

---

0100 1100
0100 1111
0100 1100
Principles of Fair Information Practice

• **Openness**
  - No programs whose existence is secret

• **Disclosure**
  - You can find out what information they know about you

• **Secondary use**
  - They can only use the data as you originally agreed

• **Correction**
  - You can correct bad information about yourself

• **Security**
  - Collectors of data responsible for securing it properly
Digital Footprint

- Digital Footprint
  - Where you go, what you do on web

  - “Data trail left by interactions in a digital environment”
  - Visits to webpages
  - What you clicked on
  - Search terms
  - Location, IP address
  - Comments made

- Allows data mining, profiling
  - Recommendations, personalisation, offers, contextual adaptation
Facebook Privacy Policy circa 2010:

When you connect with an application or website it will have access to General Information about you. The term General Information includes your and your friends' names, profile pictures, gender, user IDs, connections, and any content shared using the Everyone privacy setting. ... The default privacy setting for certain types of information you post on Facebook is set to “everyone”. ... Because it takes two to connect, your privacy settings only control who can see the connection on your profile page. If you are uncomfortable with the connection being publicly available, you should consider removing (or not making) the connection.
Edward Snowden

- Revealed widespread operations of the NSA to collect data on everyone in the world
- Revealed that government is collecting metadata from major phone companies
- NSA PRISM, XKeyscore, Tempora, Evil Olive programs
  - Direct access to Microsoft, Google, Facebook, Apple servers
  - Hacked internet backbones in Beijing, Hong Kong
  - Britain's GCHQ involved
  - 1 billion cell calls tracked every day

Hero or enemy?
• Lecture 4
  – Government transparency
  – Surreptitious versus not recording
  – Basic processing
  – Color mixing
N.Y.C. Police Maligned Paradegoers on Facebook

They called people “animals” and “savages.” One comment said, “Drop a bomb and wipe them all out.”

Hearing New York police officers speak publicly but candidly about one another and the people they police is rare indeed, especially with their names attached. But for a few days in September, a raw and rude conversation among officers was on Facebook for the world to see — until it vanished for unknown reasons.
Contrast: Surreptitious vs. Not

● Star Wars Kid
  - Ghyslain Raza
  - 2002, Messing around in drama studio in HS
  - Other kids found and uploaded
  - http://www.youtube.com/watch?v=HPPj6vilBmU

● Numa Numa Guy
  - Gary Brolsma
  - Lip-syncing at home
  - http://www.youtube.com/watch?v=W9m9UFhEARg
Coding is all about DETAIL

Two functions defined, one form

```cpp
void name () {
    ... stuff ...
}
```

Every symbol matters!

Simple statements end with semicolon (;)

Editor colors things to help you out

Some functions include things inside parentheses; called *arguments*

Arguments have meaning based on their position:

```cpp
size(width, height);
stroke(red, green, blue);
```

Editor highlights matching braces + parens

Case matters
Mixing up purple

- Colors on computers are usually given as three numbers: R G B, red green blue
- Every possible color is a mixture
  - Purple is 128, 0, 128
- The numbers are intensities of tiny lights that make up a pixel on the screen
  - Smallest intensity is 0
  - Greatest intensity is 255
  - Amazingly, R + G + B = white
• Lecture 5
  – Binary
  – Binary math
  – Advanced truth tables
  – Logic gates
  – Variables
Positional Notation

- Binary numbers use *place notation* just like decimal numbers
  - Decimal
    \[1110 = 1 \times 1000 + 1 \times 100 + 1 \times 10 + 0 \times 1\]
    \[= 1 \times 10^3 + 1 \times 10^2 + 1 \times 10^1 + 0 \times 10^0\]
  - Binary
    \[1110 = 1 \times 8 + 1 \times 4 + 1 \times 2 + 0 \times 1\]
    \[= 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0\]

1110 in binary is 14 in decimal
Binary Addition

\begin{array}{c}
001111 \\
+ 010101 \\
\hline \\
011010 \\
\end{array}
(NOT P) OR (NOT Q) versus NOT (P AND Q)

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>NOT P</th>
<th>NOT Q</th>
<th>(NOT P) OR (NOT Q)</th>
<th>P AND Q</th>
<th>NOT (P AND Q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
More logic gates!!!
What is a variable?

- A variable is a named location in the computer's memory

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>mouseX</td>
<td>167</td>
</tr>
<tr>
<td>mouseY</td>
<td>203</td>
</tr>
<tr>
<td>width</td>
<td>100</td>
</tr>
<tr>
<td>height</td>
<td>200</td>
</tr>
<tr>
<td>x</td>
<td>58</td>
</tr>
<tr>
<td>userName</td>
<td>Nathan</td>
</tr>
</tbody>
</table>
Topics

• Lecture 6
  – Digital "documents" is a metaphor
  – Authenticity is tricky
  – Artificial reality
  – Compression
  – Steganography
• PDF blackout
• Microsoft Office track changes
  – UN report on assassination
  – SCO lawsuit
  – countless resumes
• Scanned vs. PDF/doc
  – not searchable/indexable
  – harder to “read” by the blind
• Security “features” to prevent unauthorized modification, copying
Can you believe what you see?
• Computers let us make images of things that don't exist.
• “Kiddie porn” is illegal, other forms are not
• What about artificial “kiddie porn”?
  – PROTECT Act of 2003
• Second Life
  – Banned “ageplay” in 2007

This cat does not exist
Compression

- Sampling rate
  - e.g. time lapse photography
- lossy or lessless
- run length encoding
- spatial conherence (solid white background)
- temporal coherence (nothing moving)
- compression trades:
  - computing time vs. storage space/bandwidth
Hiding an image inside an image
Topics

• Lecture 7
  – Encoding text (non-ASCII)
  – Writing a book with one eyelid
  – Huffman Coding
  – Building Huffman trees
Encoding (Example)

000001 = 1
000100 = 2
001000 = 4
010000 = 8
100000 = 16
110001 = 16+8+1
      = 25

a 1   h 8   o 15   v 22
b 2   i 9   p 16   w 23
c 3   j 10  q 17  x 24
d 4   k 11  r 18  y 25
e 5   l 12  s 19  z 26
f 6   m 13  t 20

go 7   n 14  u 21

Message: 01101 10101 10011 01001 00011
• Jean-Dominique Bauby
  – Editor-in-chief of French Elle
  – (Dec. 1995) Stroke and coma
  – (Jan. 1996) Wakes up paralyzed except for left eyelid
  – Writes book *The Diving Bell and the Butterfly* using left eyelid
  – (Mar. 6, 1997) Book published, sells millions
  – (Mar. 9, 1997) Dies of pneumonia
Huffman Example

“dog cat cat bird bird bird bird fish”

<table>
<thead>
<tr>
<th>Symbol</th>
<th>dog</th>
<th>cat</th>
<th>bird</th>
<th>fish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>1/8</td>
<td>2/8</td>
<td>4/8</td>
<td>1/8</td>
</tr>
<tr>
<td>Original encoding</td>
<td>00</td>
<td>01</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>Huffman encoding</td>
<td>110</td>
<td>10</td>
<td>0</td>
<td>111</td>
</tr>
</tbody>
</table>

Expected size:
- Original: \(\frac{1}{8} \times 2 + \frac{2}{8} \times 2 + \frac{4}{8} \times 2 + \frac{1}{8} \times 2 = 2\) bits/symbol
- Huffman: \(\frac{1}{8} \times 3 + \frac{2}{8} \times 2 + \frac{4}{8} \times 1 + \frac{1}{8} \times 3 = 1.75\) bits/symbol
Building the Tree

- The algorithm (recipe)
  - Place each symbol in leaf
    - Weight is symbol frequency
  - Pick two lowest weight trees L and R (initially leaves)
  - Create new internal node
    - Left child L
    - Right child R
    - Weight on new node is sum of L and R
  - **Repeat** until there is just one tree
• Lecture 8
  – Boolean expressions
  – Statements
  – If/else statements
  – for loops (intro)
• Any expression that evaluates to true or false is a *boolean expression*.

• Making boolean expressions:
  – Relational operators <, <=, >, >=
  – Equality operators ==, !=

• Example:

```java
int i = 2;
int j = 5;

i < j
i == j
(j + 2) <= 10
```
Non-statements

• Not all expressions can be considered statements.
• Syntax errors:
  
x+y;
  width > 20;
  – These don't make sense as statements. They don't *do* anything.
• Statements must *do* something
  – assign a new value to a variable (x = 5;)
  – cause output to occur (println(), rect())
  – change some internal state (background())
The if else statement

```java
if (BooleanExpression)
    Statement
else
    Statement
```

Diagram:
- Input: BooleanExpr
  - True: Statement
  - False: Statement
Repetition

- Repeating commands is powerful
  - Lightbot 2.0 used recursion
    - function calls itself
  - Symbolic Lightbot used a prefix number
    - 4:Step means do Step Step Step Step
  - Processing uses a `for` loop

```cpp
void setup() {
  size(500, 200);
  background(0);
  fill(255);
  for(int i = 0; i < 16; i++) {
    ellipse(100 + 25*i, 100, 15, 15);
  }
}
```
• Lecture 10
  – Pair programming
  – Modern art
  – Linear interpolation of color
  – for loop (parts)
  – Functions in Processing
Why pair programming?

- Having a partner gives you a helper
- Many people find computers more fun when working with others
- Learning to work with computers is similar to learning a foreign language
  - lots of new words, codes
  - easier to learn if you have to “converse” in the language
- Working with a partner develops teamwork skills
- Real-life programming is almost always done in teams
- Employers are looking for team players
If we generate it with Processing, is it art?

Piet Mondrian (1872-1944)   Jackson Pollock (1912-1956)

Josef Albers (1888-1976)
lerpColor()

color left = color(255, 210, 10);
color right = color(255, 0, 255);
color mix;

void setup() {
  size(400, 200);
  background(0);
  noStroke();
  fill(left);
  rect(0, 0, 100, 200);
  mix = lerpColor(left, right, 0.33);
  fill(mix);
  rect(100, 0, 100, 200);
  mix = lerpColor(left, right, 0.66);
  fill(mix);
  rect(200, 0, 100, 200);
  fill(right);
  rect(300, 0, 100, 200);
}

lerp is short for linear interpolation
Here are the parts of a `for` loop (all are required)

```plaintext
for (int i = 0; i < 16; i++) {
    ellipse(100 + 25*i, 100, 15, 15);
}
```

Result: we get 16 copies of the ellipse Pac-Man pills!
Functions

- Here is the form of a function definition
  
  ```
  <return type> <name> ( <param list> ) {
    <body>
  }
  ```

- Example:
  
  ```
  void draw_box(int x_pos, int y_pos) {
    rect(x_pos, y_pos, 20, 20);
  }

  color pinkish() {
    return color(255, 200, 180);
  }
  ```
Topics

Lecture 11

- Arguments become parameters
- Function calls and function definition
- Tower of functions
If the **definition** has \( n \) parameters

- The **call** needs \( n \) arguments

The parameters and arguments correspond

```java
void draw() {
    drawTarget(width*0.25, height*0.4, 200, 4);
    drawTarget(width*0.5, height*0.5, 300, 10);
    drawTarget(width*0.75, height*0.3, 120, 6);
}

void drawTarget(float xloc, float yloc, int size, int num) {
    float grayvalues = 255/num;
    float steps = size/num;
    for (int i = 0; i < num; i++) {
        fill(i*grayvalues);
        ellipse(xloc, yloc, size - i*steps, size - i*steps);
    }
}
```
Using Functions

- Once defined, you can call functions repeatedly... that's the point of writing them!

```cpp
void setup() {
  size(120, 100);
  background(0);
  noStroke();
}

void draw() {
  fill(255);
  hexa(20, 20);
  hexa(50, 20);
  hexa(80, 20);
}

void hexa(float xbase, float ybase) {
  rect(xbase, ybase + 10, 20, 40);
  triangle(xbase, ybase + 10,
           xbase + 20, ybase + 10,
           xbase + 10, ybase);
  triangle(xbase, ybase + 50,
           xbase + 20, ybase + 50,
           xbase + 10, ybase + 60);
}
```
Review of what we did:

The program is ONLY drawing triangles and rectangles, but we don't think of it that way... to us it is a timer.

Our tower of functions!
• Lecture 12
  – Tags and metadata
  – HTTP
  – Rules of HTML
Encoding Information: There's more

- Bits and bytes encode information, but that's not all
- **Tags** encode format and some structure in word processor documents
- **Tags** encode format and some structure in HTML
- **Tags** are one form of metadata
- **Metadata** is information about information
• Web uses http:// protocol
• Browser asks for a web page
  – Usually expressed in hyper-text markup language, HTML
    • Hyper-text means you can have links that leave linear stream of text, let user leave and come back
    • Markup language is a notation to describe how a published document looks
      – fonts
      – text color
      – heading
      – images
      – page layout
Basics of HTML #1

• Rule 1
  – Content is given directly; anything not content is given inside tags

• Rule 2
  – Tags are made of < and > used like this:
    <p style="color:red">This is a paragraph</p>
    Start tag                     Content                     End tag

• Rule 3
  – Tags must be paired (start-end) or "self-terminated"
Topics

• Lecture 13
  – Catalog indexing
  – Search engines
  – Web search steps
  – Crawling
  – Precision and recall
  – Page rank
Not Just Cows

A guide to resources on the Internet and BITNET in agriculture and related subjects. Compiled by Wilfred (Bill) Drew.

Access via:
ftp ftp.sura.net; login anonymous; cd pub/nic; get agricultural.list

Information:
Email: drewwe@snymorva.bitnet
Library versus the Information Bazaar

- "Yellow Pages", directories, catalogs
- The web is not hierarchical
  - no structure like a library
- Catalogues are **out**, search engines are **in**
- **BUT**
  - Search engines control what you see
  - Track what you look for
Web Search: How does it work?

1. Gather information
2. Keep copies
3. Build an index
4. Understand the query
5. Determine the relevance of results to query
6. Determine ranking of relevant results
7. Present the results
• *Spiders* or *crawlers* wander the Web building indexes
  – Follow links
• Estimated that 0.02% to 3% of information on Web is indexed
• How often does a page get visited?
  – some frequently, others rarely
  – Crawler tracks which pages have frequent updates
• What about loops? Sensitive locations? Passwords?
  – Spiders and crawlers designed to "play nice" with servers
• *Recall* is percentage of relevant documents that are returned by the search

• *Precision* is the percentage of retrieved results that are relevant

• Results with 100% recall (perfect recall) include all relevant documents
  – but might also include documents that aren't relevant

• Results with 100% precision (perfect precision) include only relevant documents
  – but might have missed other relevant documents
Page rank algorithms

- Ranking algorithms are regarded as "crown jewels" for search companies
- Factors include:
  - keywords in headings and title
  - keywords in body text
  - site is "trustworthy" (Is ucsc.edu trustworthy?)
  - links on page are to relevant locations
  - links to this page are relevant
  - age of the page
  - quality of text
Lecture 14
- Shared secret key
- Double lock protocol
- Public key encryption, signatures
- Bitcoin is like TF2 hats
- How Bitcoin works
- Dogecoin
Shared Secret Key

• Alice and Bob get together ahead of time, both get a copy of key (sparkly)

• Protocol
  − Alice locks with sparkly key
  − Sends locked package across untrusted network
  − Bob gets locked package
  − Bob unlocks with sparkly key
Double Lock Protocol

- Protocol:
  - Alice locks with sparkly key lock (*singly locked*)
  - Alice sends to Bob
  - Bob locks with teal key lock (*double locked*)
  - Bob sends back to Alice
  - Alice unlocks sparkly key lock (*singly locked*)
  - Alice sends back to Bob
  - Bob unlocks teal key lock (*unlocked*)
Encryption Summary

- **Shared key encryption**
  - Turn messages into data that only makes sense if the recipient has the key
  - Sender and receiver use same key to scramble/unscramble the message

- **Public key encryption**
  - Everyone has public/private keys
  - Encrypt with public key of recipient to send them a message (only they can decrypt message)
  - Sign messages with private key (anyone can verify the signature with public key)

- **Designing good protocols is tricky!**
  - Lots of ways to trick people (man-in-the-middle example)
Bitcoins are like hats...

- http://www.youtube.com/watch?v=E_9R45RLNR0
How does it work?

- Based on public/private keys
  - encryption, digital signatures
- Users request to send money to another address
  - sign request
  - post request, everyone else sees it, verify it is ok
  - once >50% of network sees it, it goes through
  - everyone keeps track of every transaction
    - verify digital signatures
    - check for double spending
    - check that sender actually has the money
    - pseudo-nonymous digital records
Dogecoin

• Just a joke, right???
  – Actually most used cryptocurrency (# transactions)
  – One of largest currencies by implied market cap
  – Fastest adopted currency... ever
  – Paper bills, strong crypto, good parameters...
Topics

- Lecture 15
  - Linear time
  - Quadratic time
  - Big O
  - Quicksort
  - Polynomial time
  - NP-Hard and NP-Complete (examples)
  - Halting problem
In computer science we say that problems whose work is proportional to input size are **linear time**

- Change color balance in a photo
  - Each pixel has to be updated
- Add two columns of numbers in a spreadsheet
  - Each row has 2 numbers to add
- Crawling the web
  - Each link has to be downloaded

Not every problem is linear...
• In computer science we say this sorting algorithm runs in quadratic time

\[ \frac{1}{2} n^2 + \frac{3}{2} n \]

- If you double the input, it will take about 4 times as long to sort (for large inputs)
- Let's try:

  For \( n = 100 \), we get \( \frac{1}{2} \cdot 10000 + \frac{3}{2} \cdot 100 = 5150 \)
  
  For \( n = 200 \), we get \( \frac{1}{2} \cdot 40000 + \frac{3}{2} \cdot 200 = 20300 \)

This \( n^2 \) part gets big quick!
The Big O

• Keeping track of every detail makes the math tedious...
  – The only part that really matters is the biggest term
  – Big O notation lets us ignore all the details except the form of the largest term
    \( O(n) \)
    \( O(n^2) \)
  – We say an algorithm "runs in time \( O(n^2) \)" or similar
    • Easier than figuring out \( \frac{1}{2} n^2 + \frac{3}{2} n \)

• So our algorithm of picking the biggest row for each position runs in time \( O(n^2) \)
  – Actually pretty slow as \( n \) gets big...
Quicksort
Polynomial Time

- **Polynomial time:**
  - Means the number of steps is a polynomial of the input size
  - Considered to be "realistic"
    - You can probably solve large polynomial time problems by using powerful computers or lots of computers connected together
- But not just linear
  - Doubling problem size can more than double running time
Related Problems

• Hard problems like this without polynomial time solutions are often *NP-Hard*
  – We won't formally define this, intuition is that they *don't* have polynomial algorithms but they seem like they maybe could...
  – Other examples
    • Best way to pack boxes of different sizes onto a container ship
    • Optimal schedule of university classes to reduce conflicts
    • Cheapest way to ship books from a warehouse to customers
    • Best locations of cell towers to cover entire regions
• The halting problem cannot be solved.
• Suppose you could solve it
  – You have a program halts that takes input P and D, tells you TRUE if P(D) would stop, FALSE if P(D) would run forever.
• Let's write a sneaky function that uses H:

```java
void sneaky(program P) {
    if (halts(P, P)) {
        for(int i = 0; i < 1; i = i + 0) {
            // Loop forever!!!
        }
    }
}
```

• What does halts(sneaky, sneaky) say?
  – If it says TRUE then sneaky(sneaky) must stop (but it loops forever)
  – If it says FALSE then sneaky(sneaky) must not stop (but it finishes easily)
• Lecture 16
  – Algorithm
  – Recursion
  – Double recursion
  – Stack overflow
  – Recursive running times
  – Arduino (programming hardware)
What's an algorithm again?

- **Algorithm** is a *precise, systematic process for an agent to produce a specified result*
- Programs are algorithms
- Five properties characterize algorithms
  - **Input specified** – tell the form and amount of input required
  - **Output specified** – tell the form and amount of output produced
  - **Definiteness** – explicit steps to do, in order
  - **Effectiveness** – operations are within agent's capabilities
  - **Finiteness** – algorithms stop and give an answer or say "none"
Recursion

- **Recursion** is when the solution to a problem depends on solutions to smaller instances of the same problem
  - Algorithms that use recursion to solve problems are called *divide-and-conquer*
- **Recursive calls**
  - A function can call itself within the body of the function
  - A function can also indirectly call itself (*f* calls *g* which calls *f* which calls *g*...)

Recursive Fibonacci Function

- **fib(3):**
  - return fib(2) + fib(1)
    - fib(2):
      - return fib(1) + fib(0)
        - fib(1):
          - return 1
        - fib(0):
          - return 0
      - fib(0) is 1
      - fib(0) is 1
      - return 1 + 1
    - fib(2) is 2
    - fib(1):
      - return 1
    - fib(1) is 1
    - return 2 + 1
  - fib(3) is 3

```c
int fib(int n) {
    if(n == 0) {
        return 0;
    }
    if(n == 1) {
        return 1;
    }
    return fib(n - 1) + fib(n - 2);
}
```

Double recursion!
Oops!

`rect(xbase, ybase, w, h);`

StackOverflowError: This sketch is attempting too much recursion.

to be called recursively (it's calling itself and going in circles),
or you're intentionally calling a recursive function too much,
and your code should be rewritten in a more efficient manner.
How many steps does it take?

- Number of steps depends on:
  - $n$ (number of items in input)
- How long to merge $n$ and $m$ elements in two lists?
  - After some analysis, it takes $n+m$ steps
- Let's call the number of steps $f(n)$ (worst case)
- We know:
  - $f(1) = 1$
  - $f(n) = f(n/2) + f(n/2) + (n/2 + n/2) = 2 \cdot f(n/2) + n$
- What function works for $f$?
  - This is a tricky fun math problem!
Programming Arduino

```cpp
void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // there is a pull up resistor on the button so the unpressed
  // state is HIGH
  // check to see if the buttonState is LOW:
  if (buttonState == HIGH) {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  } else {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  }
}
```

Done compiling.

Binary sketch size: 752 bytes (of a 4096 byte maximum)
End