Bits and Bytes

How do computers compute?
All data can be represented with:
- 1s and 0s
- on/off
- true/false

Numbers?
Binary Numbers

- Five volunteers...
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
• What is the decimal equivalent of the binary number 1001?
  A. 8
  B. 9
  C. 10
  D. 11
  E. 12
Decimal to Binary

- What is the binary equivalent of the decimal value 22?
  A. 10010
  B. 10100
  C. 10101
  D. 10110
  E. 10111
Binary Combinations

• One bit
  – 0, 1

• Two bits
  – 00, 01, 10, 11

• Three bits
  – 000, 001, 010, 011, 100, 101, 110, 111

• Four bits
  – 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111
  – 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111
Binary Addition

001111
+010101

---

1000100
Binary Multiplication

111
×101

---
| ASCII | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|       | N | U | S | H | X | E | T | E | Q | A | K | B | L | R | S | H | T | L | F | Y | T | F | F | C | R | S | 0 | S | 1 |   |
|       | D | L | D | 1 | D | 2 | D | 3 | D | 4 | N | K | S | Y | E | 2 | C | N | E | M | S | B | E | C | F | S | G | S | R | S | U | S | |
| 0010  | ! | " | # | $ | % | & | ' | ( | ) | * | + | , | - | . | / |       |
| 0011  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; | < | = | > | ? |       |
| 0100  | @ | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |       |
| 0101  | P | Q | R | S | T | U | V | W | X | Y | Z | [ | \ | ] | ^ | _ |       |
| 0110  | \ | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o |       |
| 0111  | p | q | r | s | t | u | v | x | y | z | { | | | } | ~ |   |
| 1000  | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ | \ |   |
| 1001  | D | C | P | 1 | P | Z | S | E | C | M | M | M | M | S | E | P | 0 | B | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |   |
| 1010  | A | o | i | c | £ | ¥ | | | S | " | © | € | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ |   |
| 1011  | ° | ± | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ | £ |   |
| 1100  | A | A | A | A | A | A | A | C | E | B | E | E | I | I | I | I | I | I | I | I | I | I | I | I | I | I | I | I | I | I | I | I | I | I |   |
| 1101  | D | N | O | O | O | O | O | O | O | X | Ø | Ü | Ù | Ù | Ù | Ü | Ü | Ü | Ü | P | ß |       |
| 1110  | å | a | å | a | å | a | å | å | å | å | å | å | å | å | å | å | å | å | å | å | å | å | å | å | å | å | å | å | å | å | å | å | å | å |   |
| 1111  | ð | ñ | ò | ò | ò | ò | ò | ò | ò | ò | ò | ò | ò | ò | ò | ò | ò | ò | ò | ò | ò | ò | ò | ò | ò | ò | ò | ò | ò | ò | ò | ò | ò |   |
Representing Pictures

```
  0 1 1 1 1 1 1 1 0
  1 0 0 0 0 0 0 0 0
  1 0 0 0 0 0 0 0 0
  0 1 1 1 1 1 1 0 0
  0 0 0 0 0 0 1 0 0
  0 0 0 0 0 0 1 0 0
  1 1 1 1 1 1 1 0 0
  0 0 0 0 0 0 0 0 0
```

```
Representing Pictures

A. 8
B. 16
C. 32
D. 64
E. 124

254

0
But how do computers work?
Basic operations

• What computers do
  – arithmetic
  – compare bits
  – read/write memory
  – jump to new instruction

• All implemented with logic gates
<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>
## Truth Table for AND

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>P AND Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
## Truth Table for **AND**

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>P AND Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### Truth Table for OR

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>P OR Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P</td>
<td>Q</td>
<td>P OR Q</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>--------</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
## Truth Table for NOT (P AND Q)

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>P AND Q</th>
<th>NOT (P AND Q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
(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>
(\text{NOT } P) \text{ OR } (\text{NOT } Q) \text{ versus } \text{NOT} (P \text{ 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>

\text{NOT} (P \text{ AND } Q) = (\text{NOT } P) \text{ OR } (\text{NOT } Q)

This is DeMorgan's Law for boolean algebra
What is $P \lor (Q \land R)$ if
- $P$ is false
- $Q$ is true
- $R$ is true?

A. True
B. False
Exclusive OR (XOR)

• Two propositions, either of which can be true or false
• XOR means exactly one of them is true
  – Can't both be false
  – Can't both be true
• Like OR but excludes case of both true
  “Do you want to go to the opera or the movie?”
<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>P OR Q</th>
<th>P XOR Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
What use is XOR?

- Binary adder

http://en.wikipedia.org/wiki/Adder_(electronics)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

A + B = CS
More logic gates!!!

[Diagram of logic gates]
• Key principle
  – Information is represented as the presence or absence of a phenomenon at a given place and time

• Phenomenon in:
  – Computers: electrical signals on a line
  – Hollerith machine: hole in punch card

• Logic gates
  – Perform operations on bits such as AND, OR, NOT
  – Can be wired into bigger circuits to do more complicated calculations
Break
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>
Variables

- Variables store/remember values
- Values can be changed
- Each variable must be declared to store a particular kind of value
  - whole number, fraction, color, image, boolean
- Should have descriptive name
  - Must start with a letter
  - May include numbers (e.g. color2)
  - May not include spaces
  - Picking good names is an art
What does the code do?

```java
void setup() {
  size(600, 600);
}
int diameter = 0;
void draw() {
  ellipse(width/2, height/2, diameter, diameter, diameter);
  diameter = diameter + 5;
}
```

• What does the above code display?
  A. nothing (a circle with diameter 0)
  B. a circle growing and shrinking
  C. a growing white circle
  D. a growing black circle
  E. nested circles growing in size
Declaring vs. Initializing vs. Assigning

```c
int xPos; // declare xPos
int xPos = 100; // declare and initialize
xPos = offset + 10; // assign new value
```
What does the above code display?

A. nothing (just black)
B. a white circle moving left on a black background
C. a white circle moving up on a black background
D. a stack of white circles growing upwards
E. a stack of white circles growing downwards
What does the code do?

```java
void setup() {
    size(600, 600);
}
void draw() {
    background(255);
    int xPos = 0;
    ellipse(xPos, height/2, 100, 100);
    xPos = xPos + 1;
}
```

• What does the above code display?
  A. a circle moving across the screen left to right
  B. a circle moving across the screen right to left
  C. a circle moving across the screen top to bottom
  D. a circle moving across the screen bottom to top
  E. a half-circle on the left edge not moving
System Variables

- `mouseX, mouseY`
- `pmouseX, pmouseY`
- `width, height`
- `frameCount`
Fixing some code

```java
void setup() {
    size(600, 600);
}

void draw() {
    fill(255 - abs(mouseX - pmouseX));
    rect(mouseX, mouseY, mouseX, mouseY);
}
```

• Try out the code
Where should `background(255);` be placed so that the sketch just shows one moving rectangle?

```java
void setup() {
    size(600, 600);
    // E: C or D
}

void draw() {
    fill(255 - abs(mouseX - pmouseX));
    rect(mouseX, mouseY, mouseX, mouseY);
    // E: C or D
}
```
• **If** the RPMs are higher than 4000 then shift into the next gear.

• Let the dough rise on the counter until it has doubled in size.

• **If** the ball reaches the bottom of the screen change its direction.
Boolean Expressions

• 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
  ```
Expressions and Statements

- *Statements* are formed by adding a semicolon at the end of certain types of expressions.
  - An assignment
    ```javascript
    area = width * height;
    ```
  - A method call
    ```javascript
    rect(...);
    ```
Not all expressions can be considered statements.

Syntax errors:

- 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())
Blocks

• Several statements can be grouped into a \textit{block} using \{\}. A block is a statement.

```java
{  
  int x = 20, y = 30, size = 40;  
  ellipseMode(CORNER);  
  fill(255, 0, 0);  
  rect(x, y, size, size);  
  fill(0, 255, 0);  
  ellipse(x, y, size, size);  
}

// x, y, and size above cannot be used here
The if statement

if (BooleanExpression)
   Statement
int count = 0;

void setup() {
    frameRate(2);
}

void draw() {
    background(120);
    int x = 20, y = 30, size = 40;
    if (count % 2 == 0) {
        ellipseMode(CORNER);
        fill(255, 0, 0);
        rect(x, y, size, size);
        fill(0, 255, 0);
        ellipse(x, y, size, size);
    }
    count = count + 1;
}
int ballX, ballDia = 50;
void setup() {
    size(400,400);
    ballX = -ballDia/2;
}
void draw() {
    background(120);
    if (ballX > width+ballDia/2)
        ballX = -ballDia/2;
    ellipse(ballX, height/2, ballDia, ballDia);
    ballX = ballX + 1;
}

A. Ball moves across until halfway off the right edge (showing half a circle)
B. Ball moves across then disappears and doesn't come back
C. Ball moves left to right then right to left after touching right edge, repeats
D. Ball moves across jumping back to left edge as soon as it touches the right edge
E. Ball moves across moving off the right edge entirely then moves in from the left edge
The if else statement

if (BooleanExpression)
    Statement
else
    Statement

BooleanExpr

True
Statement

False
Statement
```cpp
int count = 0;
void draw() {
    background(120);
    int x = 20, y = 30, size = 40;
    ellipseMode(CORNER);
    if (count % 2 == 0) {
        fill(255, 0, 0);
        rect(x, y, size, size);
        fill(0, 255, 0);
        ellipse(x, y, size, size);
    }
    else {
        fill(0, 255, 0);
        rect(x, y, size, size);
        fill(255, 0, 0);
        ellipse(x, y, size, size);
    }
    count = count + 1;
}
```
Fixing the code

Which code goes into the blank line to make the ball move up and down, reversing direction when it hits the edges?

```java
int ballY, ballDia = 50, speed = 1;
void setup() {
    size(400, 400);
    ballY = ballDia/2;
}
void draw() {
    background(120);
    if (ballY > height-ballDia/2) {
        speed = -1;
    } else {
        speed = 1;
    }
    ellipse(width/2, ballY, ballDia, ballDia);
    ballY = ballY + speed;
}
```

A. Leave blank
B. if (ballY > ballDia/2)
C. if (ballY < ballDia/2)
D. (ballY > ballDia/2)
E. (ballY < ballDia/2)

• Which code goes into the blank line to make the ball move up and down, reversing direction when it hits the edges?
if (posX < 0)
posX = 50;

if (posX < 0) {
    posX = 50;
    posY = posY + 20;
}

if (posX < 0);
posX = 50;
posY = posY + 1;
Logical Operators

- Operators that take boolean values as inputs
- $x \land \land y$
  - is true if $x$ and $y$ are both true ($x$ AND $y$)
- $x \lor \lor y$
  - is true if $x$ or $y$ are true or both ($x$ OR $y$)
- $!x$
  - is true if $x$ is false (NOT $x$)
```java
void setup() {
  size(200, 200);
  rectMode(CORNERS);
}

int boxLeft = 50, boxRight = 150,
    boxTop = 50, boxBottom = 150;

void draw() {
  if (mouseX > boxLeft && mouseX < boxRight &&
      mouseY > boxTop && mouseY < boxBottom)
    fill(255, 0, 0);
  else {
    fill(0, 255, 0);
  }
  rect(boxLeft, boxTop, boxRight, boxBottom);
}
```
• Simulating gravity
  - $\text{pos}_{t+1} = \text{pos}_t + \text{velocity}$
  - $\text{velocity}_{t+1} = \text{velocity}_t + \text{acceleration}$
  - Gravity is a constant acceleration downwards
Gravity

• (Demo)
if (BooleanExpression) {
    Statements
}
else {
    Statements
}

Relational operators
x < y
x <= y
x > y
x >= y
x == y
x != y

Boolean operators
p && q
p || q
!p
End