

# Computer Science 50

Introduction to Computer Science I

Harvard College

Week 0

David J. Malan  
malan@post.harvard.edu

0

# Roll Call

- 1) Stand up.
- 2) Think to yourself "I am #1".
- 3) Pair up with someone; add your numbers together; take that sum as your new number.
- 4) One of you should sit down.
- 5) GOTO step 3 if still standing.

1

# Divide and Conquer



2

# Lolcats



3

## Who

I'm among "those less comfortable"		30.1%
I'm among "those more comfortable"		20.3%
I'm somewhere in between		49.6%

4

## "Those Less Comfortable"



Image from [stevens.senate.gov](http://stevens.senate.gov).

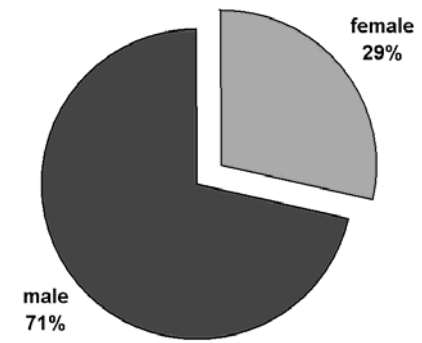
5

## Prior Programming Experience

none		43.6%
a little		46.3%
a lot		10.1%

6

## Gender



7

# Should I?

- :: Today (9/15), 3:30 – 5, Emerson 105
- :: Wednesday (9/17), 3:30 – 5, Emerson 105
- :: Thursday (9/18), 3:30 – 5, Emerson 105

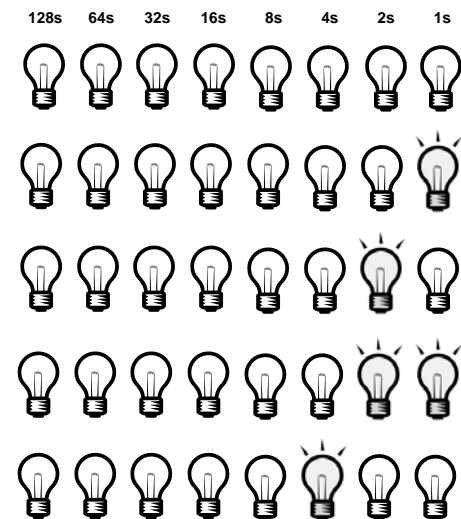
0



1



# Counting in Binary



# Counting in Binary

128s	64s	32s	16s	8s	4s	2s	1s
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	0
0	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0

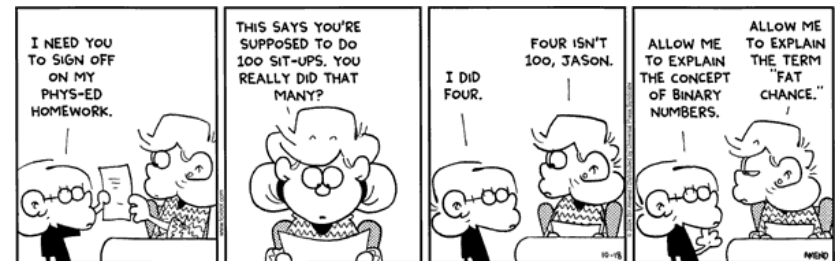
# Cambridge in Binary

128s	64s	32s	16s	8s	4s	2s	1s
---	---	---	---	---	---	---	---
---	---	---	---	---	---	---	---
---	---	---	---	---	---	---	---
---	---	---	---	---	---	---	---
---	---	---	---	---	---	---	---

# Cambridge in Binary

128s	64s	32s	16s	8s	4s	2s	1s
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0

# FoxTrot in Binary



# ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Htmi	Chr	Dec	Hx	Oct	Htmi	Chr	Dec	Hx	Oct	Htmi	Chr
0	0	000	NUL (null)	32	20	040	0x20	Space	64	40	100	0x64	@	96	60	140	0x96	^
1	1	001	SOH (start of heading)	33	21	041	0x21	!	65	41	101	0x65	A	97	61	141	0x97	a
2	2	002	STX (start of text)	34	22	042	0x22	"	66	42	102	0x66	B	98	62	142	0x98	b
3	3	003	ETX (end of text)	35	23	043	0x23	#	67	43	103	0x67	C	99	63	143	0x99	c
4	4	004	EOT (end of transmission)	36	24	044	0x24	\$	68	44	104	0x68	D	100	64	144	0x100	d
5	5	005	ENQ (enquiry)	37	25	045	0x25	%	69	45	105	0x69	E	101	65	145	0x101	e
6	6	006	ACK (acknowledge)	38	26	046	0x26	&	70	46	106	0x70	F	102	66	146	0x102	f
7	7	007	BEL (bell)	39	27	047	0x27	'	71	47	107	0x71	G	103	67	147	0x103	g
8	8	010	BS (backspace)	40	28	050	0x40	(	72	48	110	0x72	H	104	68	150	0x104	h
9	9	011	TAB (horizontal tab)	41	29	051	0x41	)	73	49	111	0x73	I	105	69	151	0x105	i
10	A	012	LF (NL line feed, new line)	42	2A	052	0x42	*	74	4A	112	0x74	J	106	6A	152	0x106	j
11	B	013	VT (vertical tab)	43	2B	053	0x43	+	75	4B	113	0x75	K	107	6B	153	0x107	k
12	C	014	FF (NP form feed, new page)	44	2C	054	0x44	,	76	4C	114	0x76	L	108	6C	154	0x108	l
13	D	015	CR (carriage return)	45	2D	055	0x45	-	77	4D	115	0x77	M	109	6D	155	0x109	m
14	E	016	SO (shift out)	46	2E	056	0x46	.	78	4E	116	0x78	N	110	6E	156	0x110	n
15	F	017	SI (shift in)	47	2F	057	0x47	/	79	4F	117	0x79	O	111	6F	157	0x111	o
16	10	020	DLE (data link escape)	48	30	060	0x48	0	80	50	120	0x80	P	112	70	160	0x112	p
17	11	021	DC1 (device control 1)	49	31	061	0x49	1	81	51	121	0x81	Q	113	71	161	0x113	q
18	12	022	DC2 (device control 2)	50	32	062	0x50	2	82	52	122	0x82	R	114	72	162	0x114	r
19	13	023	DC3 (device control 3)	51	33	063	0x51	3	83	53	123	0x83	S	115	73	163	0x115	s
20	14	024	DC4 (device control 4)	52	34	064	0x52	4	84	54	124	0x84	T	116	74	164	0x116	t
21	15	025	NAK (negative acknowledge)	53	35	065	0x53	5	85	55	125	0x85	U	117	75	165	0x117	u
22	16	026	SYN (synchronous idle)	54	36	066	0x54	6	86	56	126	0x86	V	118	76	166	0x118	v
23	17	027	ETB (end of trans. block)	55	37	067	0x55	7	87	57	127	0x87	W	119	77	167	0x119	w
24	18	030	CAN (cancel)	56	38	070	0x56	8	88	58	130	0x88	X	120	78	170	0x120	x
25	19	031	EM (end of medium)	57	39	071	0x57	9	89	59	131	0x89	Y	121	79	171	0x121	y
26	1A	032	SUB (substitute)	58	3A	072	0x58	:	90	5A	132	0x90	Z	122	7A	172	0x122	z
27	1B	033	ESC (escape)	59	3B	073	0x59	;	91	5B	133	0x91	[	123	7B	173	0x123	{
28	1C	034	FS (file separator)	60	3C	074	0x60	<	92	5C	134	0x92	\	124	7C	174	0x124	
29	1D	035	GS (group separator)	61	3D	075	0x61	=	93	5D	135	0x93	]	125	7D	175	0x125	}
30	1E	036	RS (record separator)	62	3E	076	0x62	>	94	5E	136	0x94	^	126	7E	176	0x126	~
31	1F	037	US (unit separator)	63	3F	077	0x63	?	95	5F	137	0x95	_	127	7F	177	0x127	DEL

Source: [www.asciitable.com](http://www.asciitable.com)

# This is CS 50.

Introduction to the intellectual enterprises of computer sciences. Algorithms: design, implementation, analysis. Software development: abstraction, encapsulations, data structures, debugging, testing. Architecture of computers: low-level data representation, instructions processing. Computer Systems: programming languages, compilers, operating systems, databases. Computers in the real world: networks, websites, security, forensics, cryptography. This course teaches students how to think more carefully and how to solve problems more effectively. Problem sets involve extensive programming in C as well as PHP and JavaScript.

No previous programming experience required.

This course, when taken for a letter grade, meets the Core area requirement for Quantitative Reasoning.

## Expectations\*

- :: Attend all lectures and sections
- :: Complete nine problem sets
- :: Take two quizzes
- :: Produce a final project

\* No final exam.

## Grades\*

- :: Problem Sets (best 8 out of 9) 60%
- :: Quizzes 30%
- :: Final Project 10%

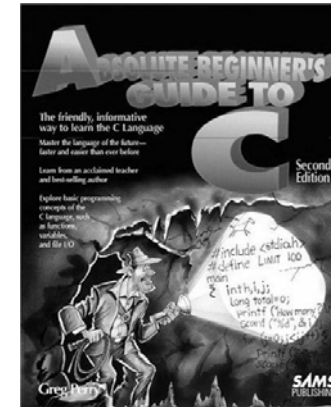
\* You may take the course pass/fail.

## Website

<http://cs50.net/>

20

## Recommended For Those Less Comfortable



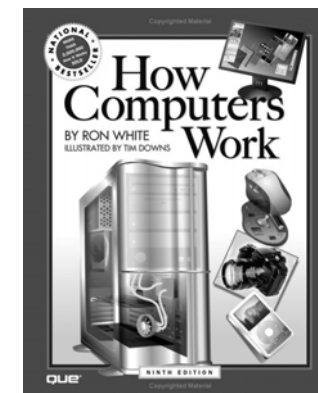
21

## Recommended For Those More Comfortable



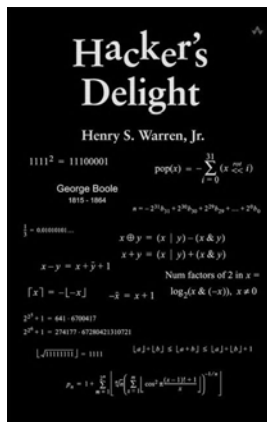
22

## Recommended For Everyone



23

## Recommended For Aspiring Hackers



24

## Lectures\*



\* "Lunch with David" on Fridays (rsvp@cs50.net)

25

## Sections

- :: For "those less comfortable"
- :: For "those more comfortable"
- :: For those somewhere in between

26

## Staff

- :: Teaching Fellows
- :: Course Assistants
- :: Virtual Teaching Fellows
- :: Sysadmins
- :: Scribes
- :: Producers, Videographers, AV
- :: me

27

# Head TF



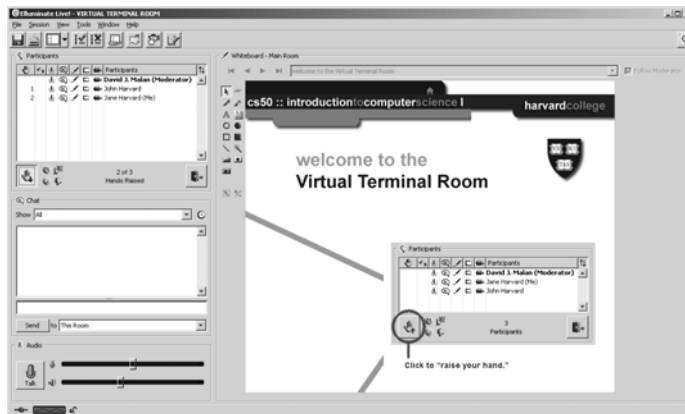
Cansu Aydede '11  
aydede@fas.harvard.edu  
+1-617-832-5239

Applied Math, Computer Science  
Eliot House

# Office Hours



# Virtual Office Hours



# Workload

0 - 5 hours		4.0%
5 - 10 hours		38.4%
10 - 15 hours		33.3%
15 - 20 hours		17.4%
20+ hours		6.9%



# Lectures

Week 0

Introduction. Bits. Binary. ASCII. Programming.  
Algorithms. Scratch. Statements. Boolean expressions.  
Conditions. Loops. Variables. Threads. Events.



32

# Lectures

Week 1

C. Source code. Compilers. Object code. SSH. SFTP.  
GCC. Functions. Comments. Standard output. Arithmetic  
operators. Precedence. Associativity. Local variables.  
Types. Casting. Standard input. Libraries. Boolean  
expressions, continued. Conditions, continued. Loops,  
continued.

```
printf("hai, world\n");
```

33

# Lectures

Week 2

Functions, continued. Global variables. Parameters.  
Return Values. Stack. Frames. Scope. Arrays. Strings.  
Command-line arguments. Cryptography.

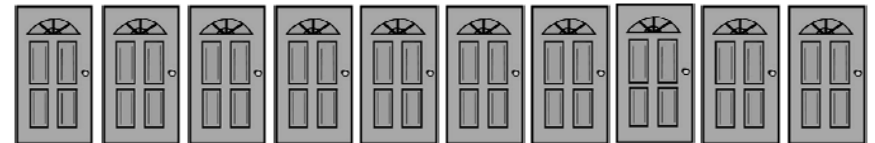


34

# Lectures

Week 3

Linear search. Binary search. Asymptotic notation.  
Recursion. Pseudorandomness. Bubble sort. Selection  
sort. Insertion sort. Merge sort.

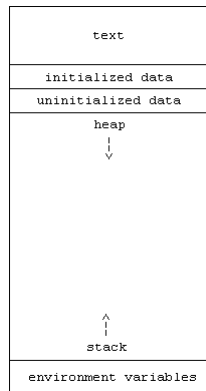


35

# Lectures

Week 4

Structures. Dynamic memory allocation. Pointers.

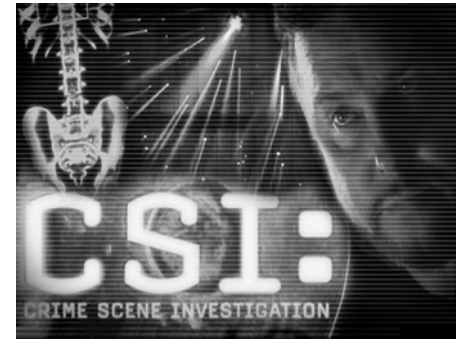


36

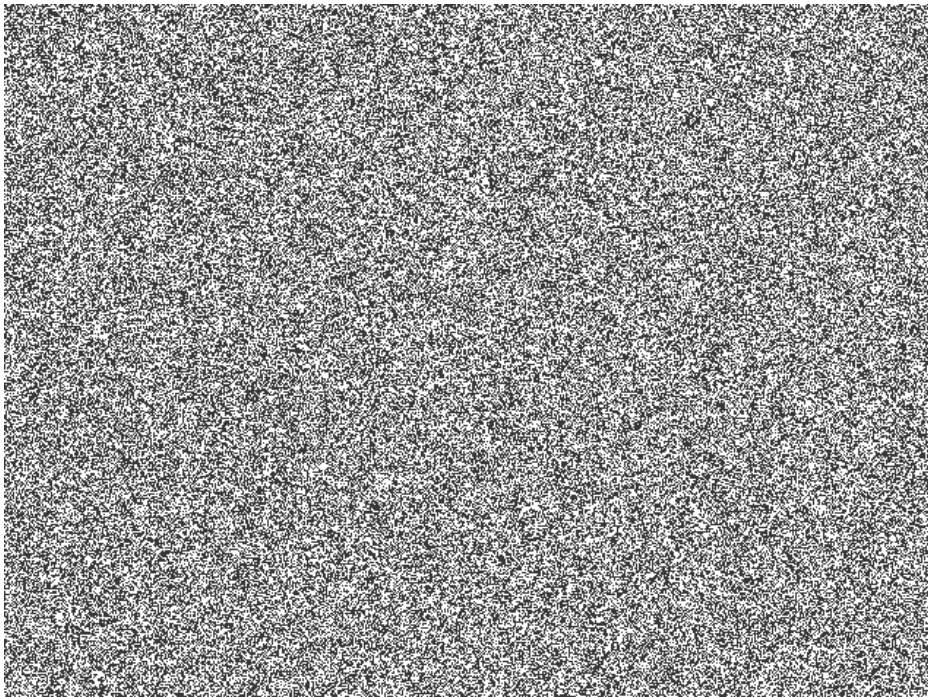
# Lectures

Week 5

Pointers, continued. Heap. Debugging. File I/O. Forensics.



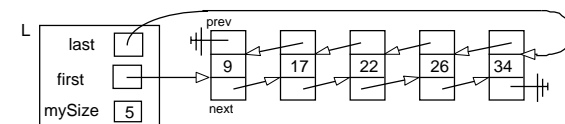
37



# Lectures

Week 6

Linked lists.



39

# Lectures

Week 7

Hash tables. Binary search trees. Huffman coding.  
Heaps. Heapsort. Tries.

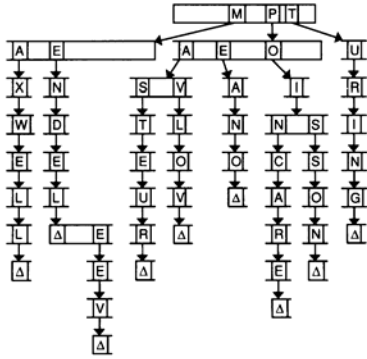


Image from *Data Structures & Their Algorithms*.

# Lectures

Week 8

TCP/IP. HTTP. XHTML. PHP. SQL.



# Lectures

Week 9

DOM. CSS. Inheritance. JavaScript. Events, continued.  
OOP. Ajax.



Image from [www.gearthblog.com](http://www.gearthblog.com).

# Lectures

Week 10

Preprocessing. Compiling. Assembling.  
Linking. CPUs.



Image from [regmedia.co.uk](http://regmedia.co.uk).

# Lectures

Week 11

Enterprise architectures. Virtualization. Cloud computing.  
Sneak previews.



Image from amazon.com.

44

# Lectures

Week 12

Exciting conclusion.



45

# Problem Sets\*

- :: Problem Set 0: Scratch
- :: Problem Set 1: C
- :: Problem Set 2: Crypto
- :: Problem Set 3: To Be Named
- :: Problem Set 4: To Be Named
- :: Problem Set 5: Forensics
- :: Problem Set 6: Misspellings
- :: Problem Set 7: XHTML + PHP + SQL
- :: Problem Set 8: JavaScript

\* Hacker Editions too

46

# Final Project\*

- :: Build something of interest to you.
- :: Make something useful.
- :: Solve an actual problem.
- :: Somehow impact campus.

\* CS 50 Fair

47

# kthxbai

# Algorithms

- 1) let socks\_on\_feet = 0
- 2) while socks\_on\_feet != 2
- 3)     open sock drawer
- 4)     look for sock
- 5)     if you find a sock then
- 6)         put on sock
- 7)         socks\_on\_feet++
- 8)         look for matching sock
- 9)         if you find a matching sock then
- 10)             put on matching sock
- 11)             socks\_on\_feet++
- 12)             close sock drawer
- 13)         else
- 14)             remove first sock from foot
- 15)             socks\_on\_feet--
- 16)     else
- 17)         do laundry and replenish sock drawer

# O Hai, C! hai.c

```
#include <stdio.h>

int
main(int argc, char * argv[])
{
    printf("O hai, world!\n");
}
```

# O Hai, C!

```
#include <stdio.h>

int
main(int argc, char * argv[])
{
    printf("O hai, world!\n");
}
```



```
10000011 00000001 00010001 00000000 00111101 11111100 01110100 00111101
00000000 01000000 00000000 00000000 00000000 00000000 00000000 00000000
10010000 00000000 00000000 00000000 01010000 00000000 00000000 00000000
00001011 00000001 00010111 00000011 00001010 00000000 00000000 00000000
00000000 00100000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00100000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01110000 00010000 00000000 00100000 00000001 00000000 00000000 00000000
00000000 00000000 00000000 00100000 00000001 00000000 00000000 00000000
00000000 00000000 00000000 01000000 00000001 00000000 00000000 00000000
00000000 00100000 00000000 01000000 00000001 00000000 00000000 00000000
11111111 11111111 11111111 11111111 11111111 11111111 11111111 11111111
10010000 10000000 00000000 01000000 00000001 00000000 00000000 00000000
00101110 01100100 01111001 01101110 01100001 01101101 01101001 01100011
10110000 00001100 00000000 00100000 00000001 00000000 00000000 00000000
10110000 00000100 00000000 00100000 00000001 00000000 00000000 00000000
10100000 00000001 00000000 00000000 00000000 00000000 00000000 00000000
10110000 00000100 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
[...]
```

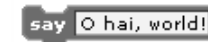
# O Hai, Scratch!

Hai1.sb



52

# Statements



53

# Statements

Hai{2,3}.sb



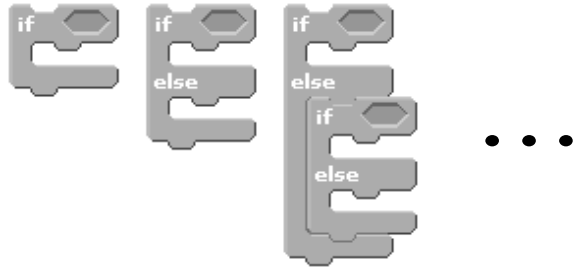
54

# Boolean Expressions



55

# Conditions



# Conditions

Hai{4,5}.sb

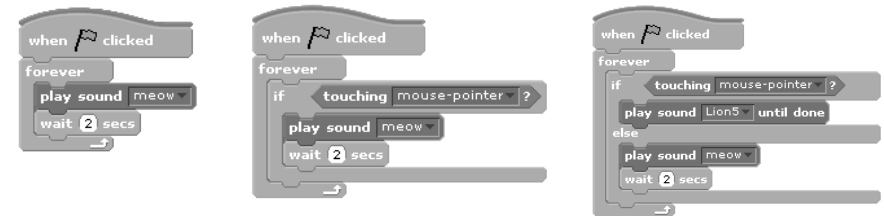


# Loops



# Loops

Hai{6,7,8}.sb



# Variables

Count{1,2}.sb

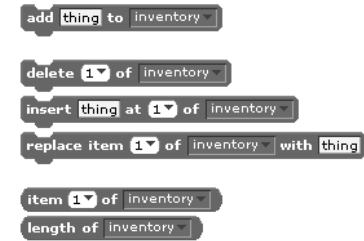


```
when clicked
  set counter to 0
  forever
    say counter for 2 secs
    wait 1 secs
    change counter by 1

when clicked
  set number to pick random 1 to 10
  say number
  if number < 6
    play sound Sheep1
```

60

# Arrays

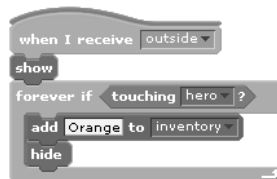


```
add thing to inventory
delete 1 of inventory
insert thing at 1 of inventory
replace item 1 of inventory with thing
item 1 of inventory
length of inventory
```

61

# Arrays

FruitcraftRPG.sb



```
when I receive outside
  show
  forever if touching hero?
    add Orange to inventory
  hide
```

62

# Threads

Move1.sb



```
when clicked
  go to x: 0 y: 0
  point in direction 90
  forever
    if touching edge?
      play sound Scream-male2 until done
      turn 180 degrees
    move 5 steps
```

63




# Threads

Move2.sb

```
when clicked
  go to x: -150 y: 150
  point in direction 45
  forever
    if touching edge?
      if on edge, bounce
    if not touching cat?
      move 3 steps
```

```
when clicked
  go to x: -160 y: -160
  point in direction pick random 91 to 179
  forever
    if touching bird?
      play sound roar
      stop script
    point towards bird
    move 1 steps
```



64

# Threads

Hai10.sb

```
when clicked
  forever
    if muted = 0
      play sound SeaLion
      think O hall for 2 secs
    wait 1 secs
```

```
when clicked
  set muted to 0
  forever
    if key space pressed?
      if muted = 0
        set muted to 1
      else
        set muted to 0
    wait 1 secs
```


65

# Threads

David.sb

```
when clicked
  set size to 70 %
  go to x: 0 y: -5
  set score to 0
  point in direction 90
  clear graphic effects
  forever
    point in direction 90
    set x to pick random -180 to 180
    wait 0.5 secs
    say [ ]
    if touching leftdove? or touching rightdove?
      say stop that
      change score by 1
      point in direction pick random 70 to 90
      change color effect by 10
    if score > 13
      point in direction 0
      say I got poned!
      stop script
```

```
when clicked
  go to x: 100 y: -140
  point in direction 0
  when down arrow key pressed
    go to front
    set y to -140
    point in direction 0
    move 60 steps
    turn 15 degrees
    play sound Glass2
    rest for 0.4 beats
    turn 15 degrees
    move -60 steps
  when left arrow key pressed
    change x by -6
  when right arrow key pressed
    change x by 6
```



66

# Events

Marco.sb

```
when clicked
  forever
    if key space pressed?
      say Marco for 2 secs
      broadcast event
```

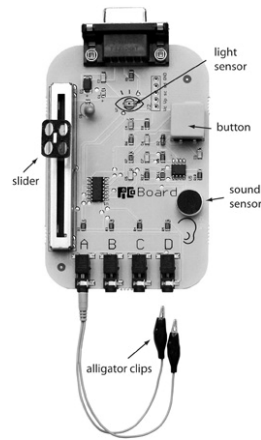
```
when I receive event
  say Polo! for 2 secs
```



67

# Sensors

singer.sb, Masquerade.sb, davidwu.sb, Electricity.sb



68

# Oscartime

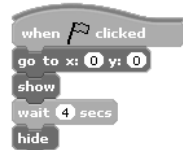
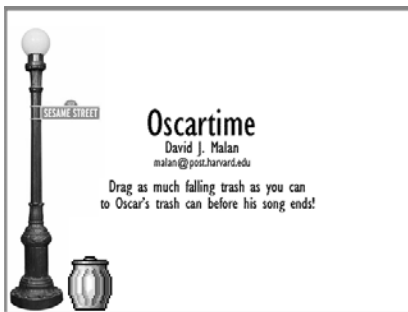
Oscartime.sb



69

# Oscartime

Displaying the Instructions



70

# Oscartime

Making Trash Fall



71

## Oscartime Implementing Dragging



```

when I receive trash_click
  forever
    if mouse down?
      go to mouse-pointer
    else
      set good_click to 5
      set my_click to 0
      stop script
  
```

72

## Oscartime Imposing a Time Limit



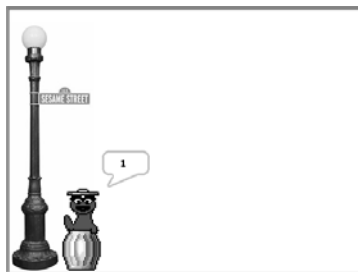
```

when clicked
  set costume to oscar1
  go to x: -140 y: -122
  show
  set playing to 1
  set score to 0
  set scoring to 0
  play sound soundtrack
  wait 134 secs
  set playing to 0
  wait 2 secs
  set costume to oscar1
  wait 0.25 secs
  set costume to oscar3
  wait 0.1 secs
  set costume to oscar4
  wait 0.1 secs
  set costume to oscar5
  wait 0.1 secs
  set costume to oscar6
  wait 1 secs
  say Your score is...
  wait 3 secs
  say score
  wait 3 secs
  say Thanks for all the trash!
  wait 5 secs
  stop all
  
```



73

## Oscartime Keeping Score



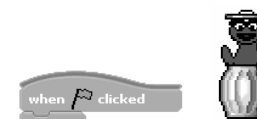
```

when I receive scored
  if playing = 1
    set scoring to 1
    wait 0.5 secs
    set costume to oscar1
    wait 0.25 secs
    set costume to oscar3
    wait 0.1 secs
    set costume to oscar4
    wait 0.1 secs
    set costume to oscar5
    wait 0.1 secs
    set costume to oscar6
    change score by 1
    say score
    wait 15 secs
    say nothing
    set costume to oscar7
    wait 0.1 secs
    set costume to oscar8
    wait 0.1 secs
    set costume to oscar1
    wait 0.1 secs
    set scoring to 0
  
```



74

## Oscartime Raising Oscar's Lid



```

when clicked
  forever
    if playing = 1 and not scoring = 1
      if distance to Trash < 40 or distance to Sneaker < 40 or distance to Newspaper < 40 or
        set costume to oscar2
      else
        set costume to oscar1
  
```

75

# Scratch Meets C



```
int
main(int argc, char * argv[])
{
    printf("O hai, world!\n");
}
```

76

# Statements

Scratch v. C



```
printf("O hai, world!\n");
```

77

# Boolean Expressions

Scratch v. C

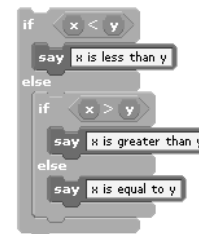


```
(x < y)
((x < y) && (y < z))
```

78

# Conditions

Scratch v. C



```
if (x < y)
{
    printf("x is less than y\n");
}
else if (x > y)
{
    printf("x is greater than y\n");
}
else
{
    printf("x is equal to y\n");
}
```

79

## Loops

Scratch v. C



```
while (1)
{
    printf("O hai!\n");
}
```



```
for (int i = 0; i < 10; i++)
{
    printf("O hai!\n");
}
```

80

## Variables

Scratch v. C



```
int counter = 0;
while (1)
{
    printf("%d\n", counter);
    counter++;
}
```

81

## Arrays

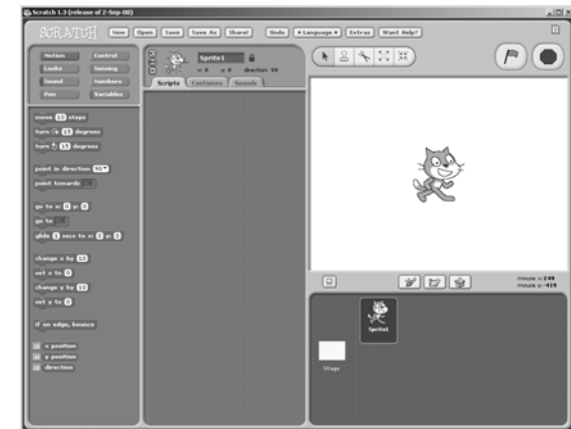
Scratch v. C



```
char *inventory[SIZE];
inventory[i] = "Orange";
```

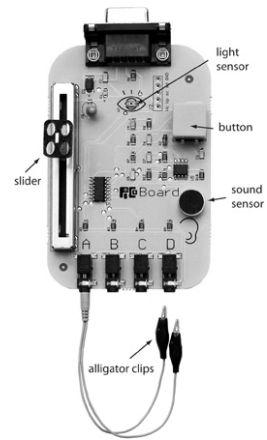
82

## Problem Set 0



83

# Problem Set 0 HACKER EDITION



84

Computer Science 50  
Introduction to Computer Science I

Harvard College

Week 0

David J. Malan  
malan@post.harvard.edu

85