



Computer Science 50

Introduction to Computer Science I

Harvard College

Week 0

David J. Malan
malan@post.harvard.edu

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.

Divide and Conquer



Lolcats



Who

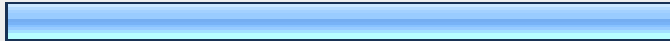
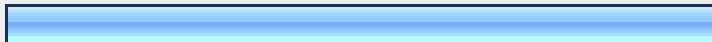
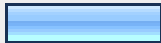
I'm among "those less comfortable"		30.1%
I'm among "those more comfortable"		20.3%
I'm somewhere in between		49.6%

“Those Less Comfortable”

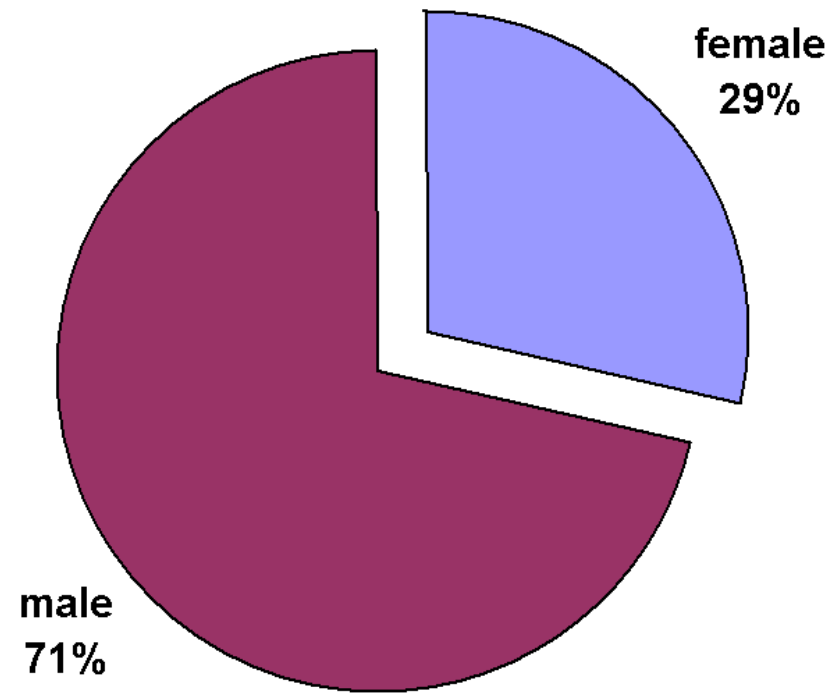


Image from [stevens.senate.gov](https://www.stevens.senate.gov).

Prior Programming Experience

none		43.6%
a little		46.3%
a lot		10.1%

Gender

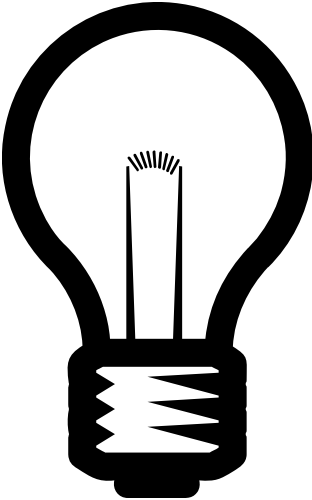


Should I?

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



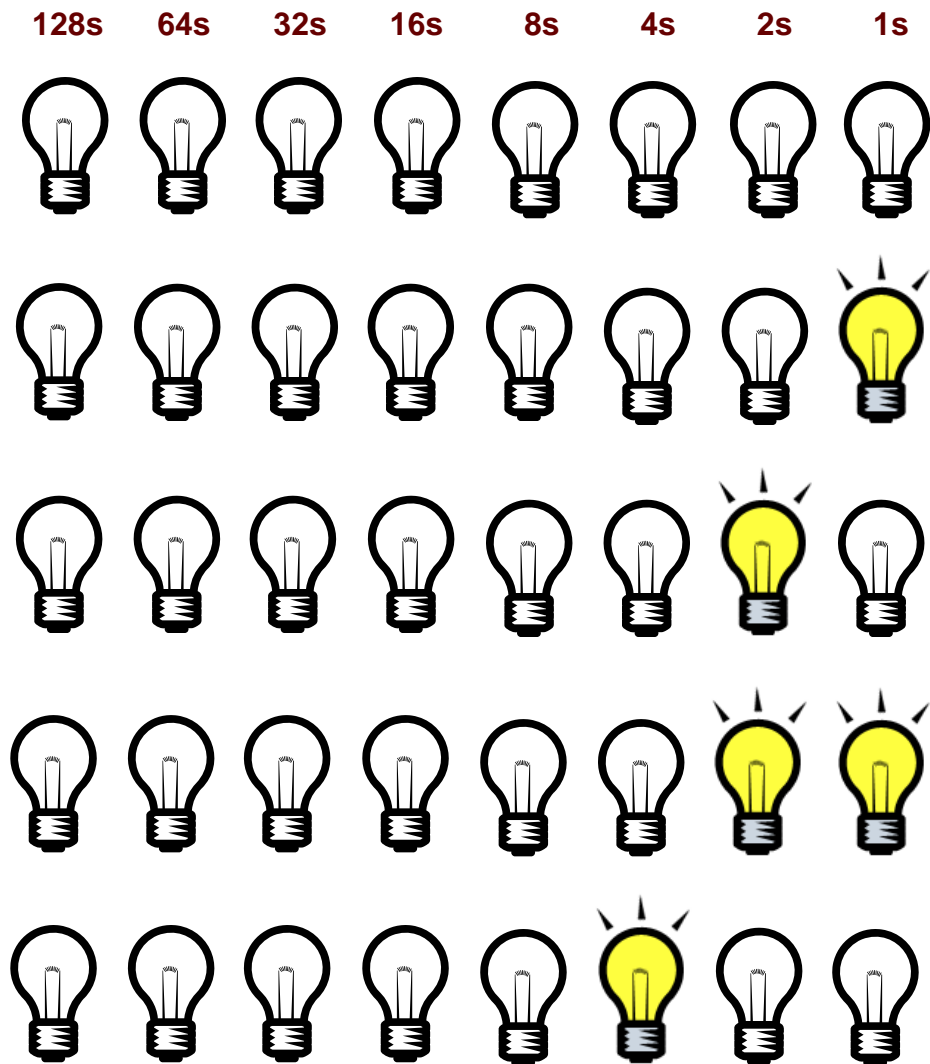
0



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

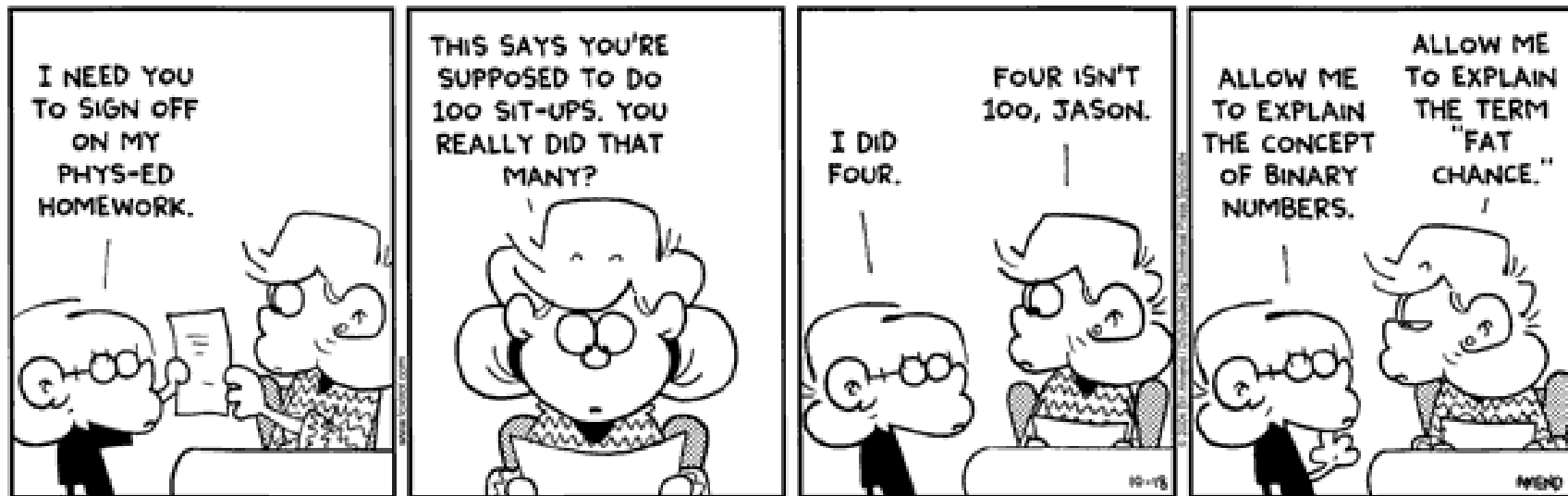
128s 64s 32s 16s 8s 4s 2s 1s

—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—

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	1	0
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	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	 	Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	\$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	((72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051))	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[[123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|	
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135]]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177		DEL

Source: 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.

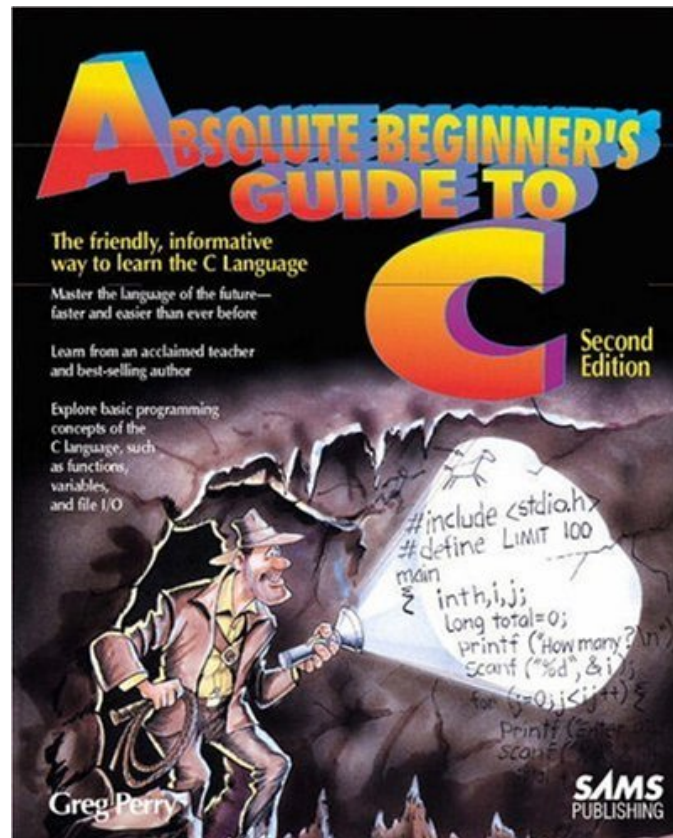


Website

<http://cs50.net/>

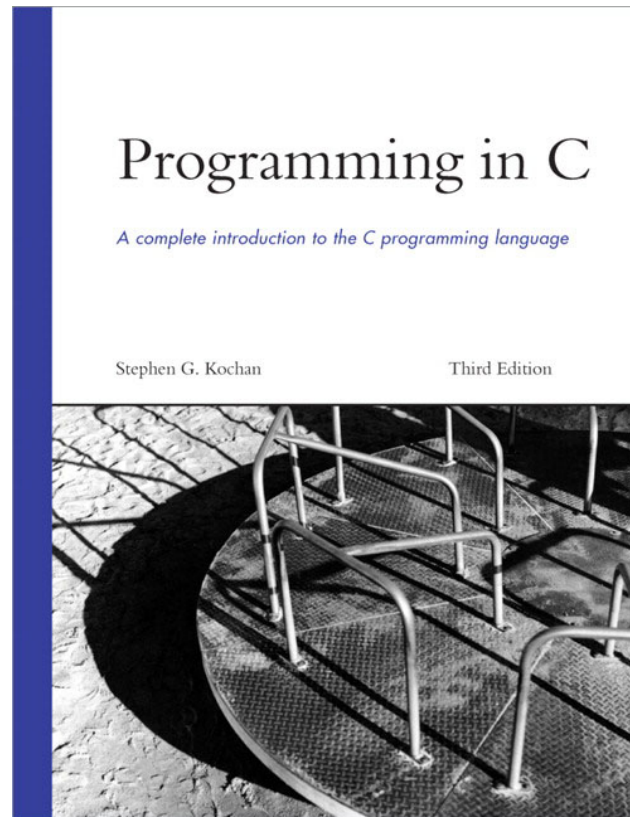
Recommended

For Those Less Comfortable

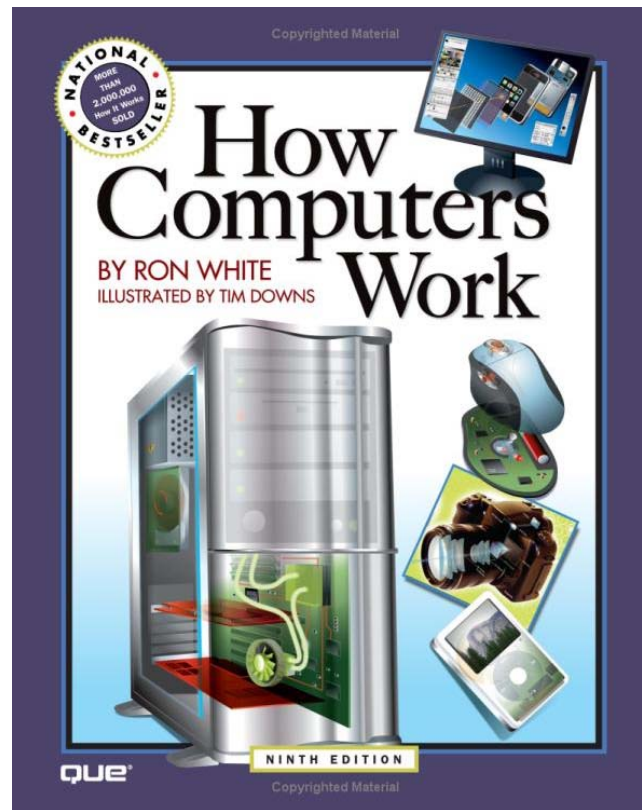


Recommended

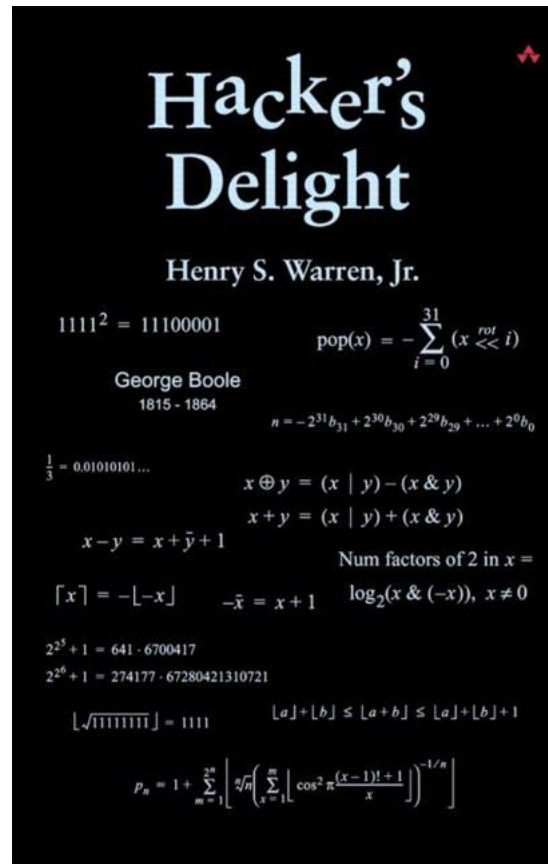
For Those More Comfortable



Recommended For Everyone



Recommended For Aspiring Hackers



Lectures*



* “Lunch with David” on Fridays (rsvp@cs50.net)

Sections

- :: For “those less comfortable”
- :: For “those more comfortable”
- :: For those somewhere in between

Staff

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

Head TF



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

Applied Math, Computer Science
Eliot House

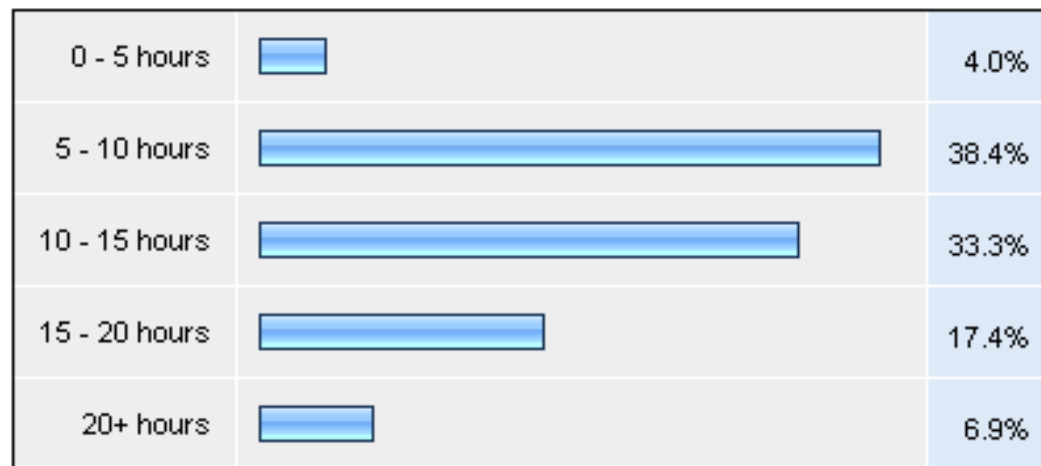
Office Hours





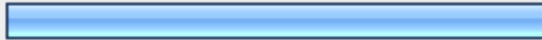
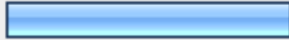

Virtual Office Hours

The screenshot displays the Elluminate Live! interface for a virtual session. The main window is titled "Whiteboard - Main Room" and contains a presentation slide. The slide has a red header with the text "cs50 :: introduction to computer science I" and "harvardcollege" on the right. The main content of the slide reads "welcome to the Virtual Terminal Room" and includes the Harvard crest. On the left side of the interface, there are several panels: a "Participants" list showing "David J. Malan (Moderator)", "John Harvard", and "Jane Harvard (Me)"; a "Chat" panel with a "Show All" dropdown and a text input field; and an "Audio" panel with "Talk" and "Mute" buttons and volume sliders. A smaller inset window, also titled "Participants", is overlaid on the whiteboard content, showing the same list of participants. A red circle highlights the "raise hand" icon in this inset, with a red arrow pointing to it and the text "Click to 'raise your hand.'" below it.

Workload



A horizontal bar chart illustrating the distribution of workload across five categories. The categories are '0 - 5 hours', '5 - 10 hours', '10 - 15 hours', '15 - 20 hours', and '20+ hours'. The bars are light blue with a thin black outline. The percentage values are displayed to the right of each bar. The '5 - 10 hours' category has the highest percentage at 38.4%, followed by '10 - 15 hours' at 33.3%.

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.



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");
```

Lectures

Week 2

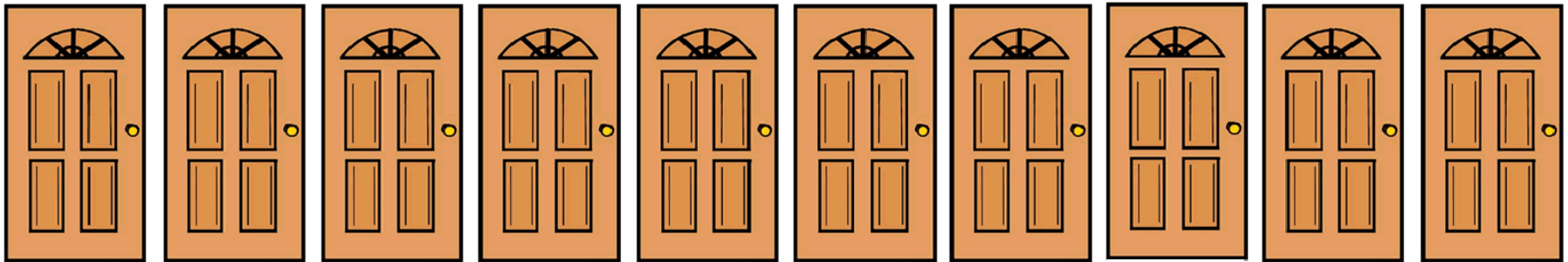
Functions, continued. Global variables. Parameters.
Return Values. Stack. Frames. Scope. Arrays. Strings.
Command-line arguments. Cryptography.



Lectures

Week 3

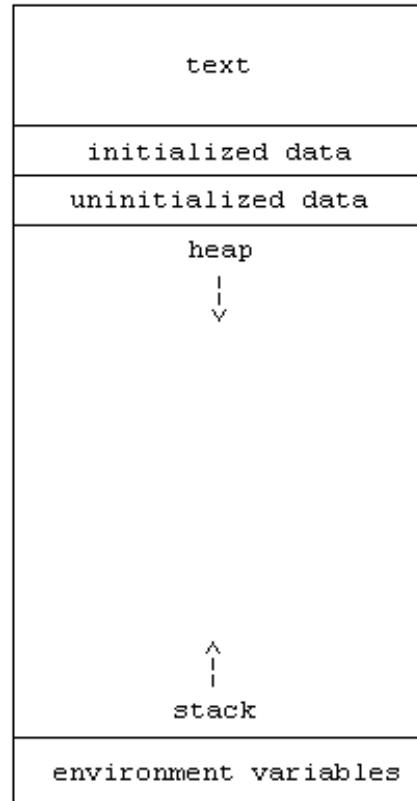
Linear search. Binary search. Asymptotic notation.
Recursion. Pseudorandomness. Bubble sort. Selection
sort. Insertion sort. Merge sort.



Lectures

Week 4

Structures. Dynamic memory allocation. Pointers.

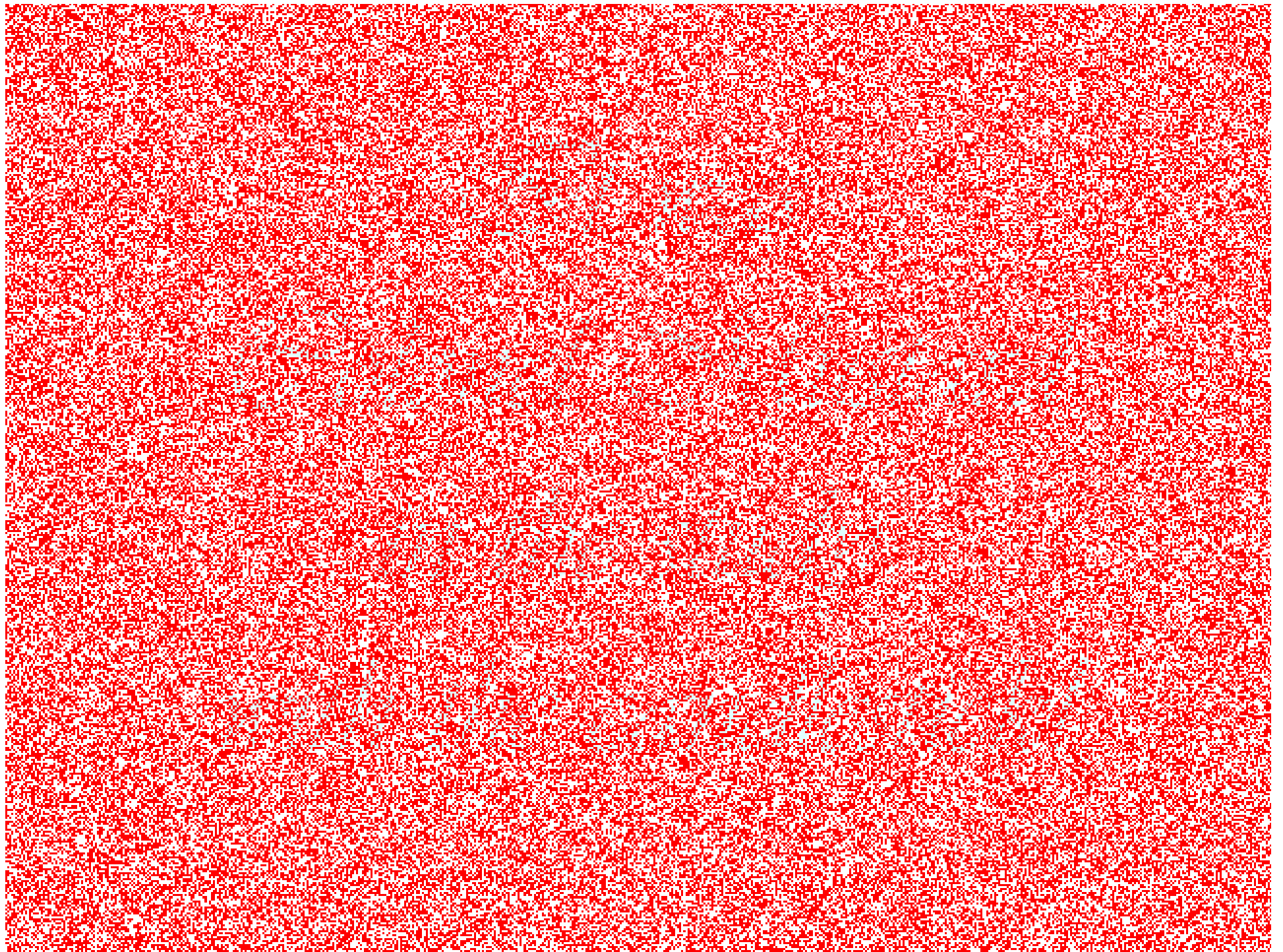


Lectures

Week 5

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

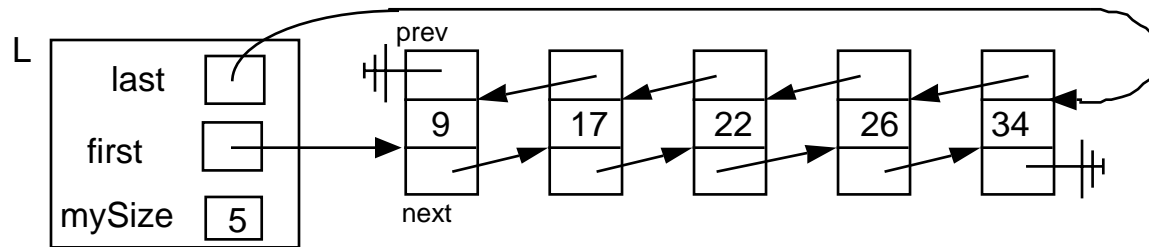




Lectures

Week 6

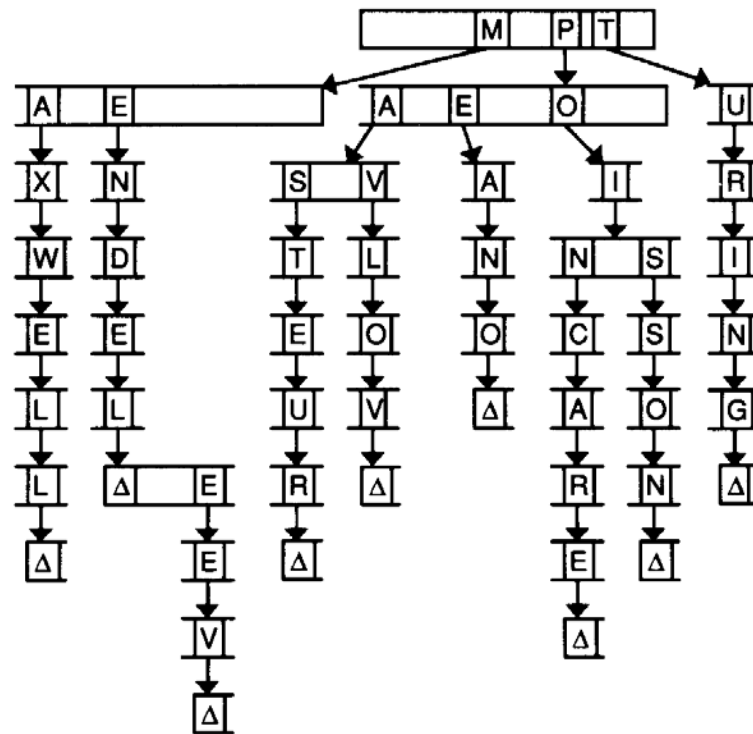
Linked lists.



Lectures

Week 7

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



Lectures

Week 8

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

The screenshot shows a web browser window displaying the Harvard College CS50 website. The browser's address bar shows the URL `http://cs50.net/index.php`. The website has a dark red header with the text "cs50 :: introduction to computerscience I" and the "harvardcollege" logo. The main content area is titled "This is CS 50." and includes a quote from David J. Malan, the instructor, describing the course as "Demanding, but definitely doable. Social, but educational. A focused topic, but broadly applicable skills. CS 50 is the quintessential Harvard course." Below the quote, there is a paragraph explaining that the course is a first course in computer science at Harvard College, designed for both concentrators and non-concentrators. A small bar chart shows the distribution of students' self-reported comfort levels with the course: 50.1% are among those less comfortable, 20.3% are among those more comfortable, and 49.6% are somewhere in between. The website also features a sidebar with navigation links such as "FREQUENTLY ASKED QUESTIONS", "VIRTUAL TERMINAL ROOM", and "Final Project". The footer contains copyright information for 2009 and mentions that the course is supported by Amazon, Google, and Microsoft.

Comfort Level	Percentage
In among "those less comfortable"	50.1%
In among "those more comfortable"	20.3%
In somewhere in between	49.6%

Lectures

Week 9

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

MONSTER MILKTRUCK!



Lectures

Week 10

Preprocessing. Compiling. Assembling.
Linking. CPUs.



Lectures

Week 11

Enterprise architectures. Virtualization. Cloud computing.
Sneak previews.



Lectures

Week 12

Exciting conclusion.



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

Final Project*

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

* CS 50 Fair

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

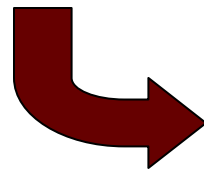
```
#include <stdio.h>

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 00000111 00110000
00001011 00000001 00001011 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 00000100 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 00100000 00000000 00000000
[...]
```

O Hai, Scratch!

Hai1.sb



Statements

say O hai, world!

wait 1 secs

play sound meow ▼

...

Statements

Hai{2,3}.sb



Boolean Expressions

touching mouse-pointer ?

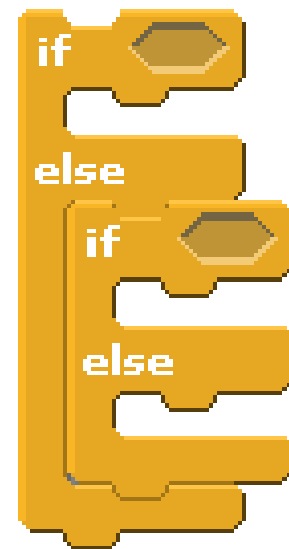
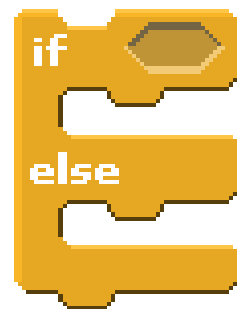
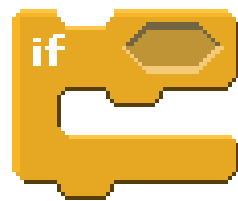
mouse down?

<

and

...

Conditions



Conditions

Hai{4,5}.sb



Loops



Loops

Hai{6,7,8}.sb

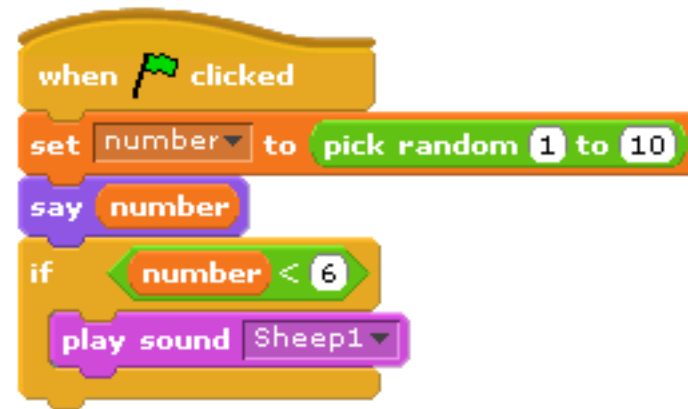
```
when clicked
  forever
    play sound meow
    wait 2 secs
```

```
when clicked
  forever
    if touching mouse-pointer
      play sound meow
      wait 2 secs
```

```
when clicked
  forever
    if touching mouse-pointer
      play sound Lion5 until done
    else
      play sound meow
      wait 2 secs
```

Variables

Count{1,2}.sb



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`

Arrays

FruitcraftRPG.sb



Threads

Move1.sb



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
```

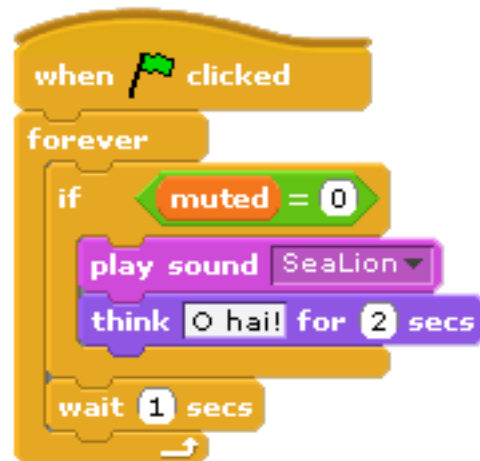


```
when clicked
  go to x: -160 y: -160
  point in direction pick random 91 to 179
  forever
    if touching bird ?
      play sound roar
      stop script
    point towards bird
    move 1 steps
```



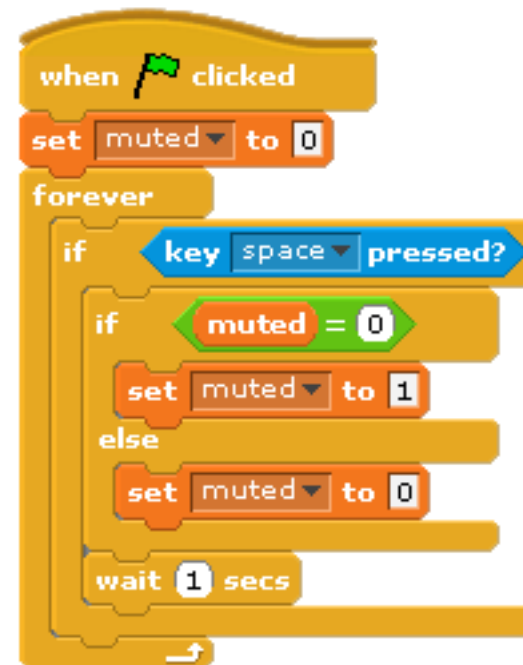
Threads

Hai10.sb



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

A Scratch code block starting with a 'when clicked' event. It contains a 'forever' loop. Inside the loop, there is an 'if' block with the condition 'muted = 0'. If true, it plays the 'SeaLion' sound and says 'O hai!' for 2 seconds. After the 'if' block, there is a 'wait 1 secs' block. The loop ends with a return arrow.



```
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
```

A Scratch code block starting with a 'when clicked' event. It first sets a variable 'muted' to 0. Then it enters a 'forever' loop. Inside the loop, there is an 'if' block with the condition 'key space pressed?'. If true, there is another 'if' block with the condition 'muted = 0'. If true, it sets 'muted' to 1. If false, it sets 'muted' to 0. After this nested 'if' block, there is a 'wait 1 secs' block. The loop ends with a return arrow.

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 leftGlove ? or touching rightGlove ?
      say stop that
      change score by 1
      point in direction pick random 70 to 90
      change color effect by 10
    if score > 15
      point in direction 0
      say I got pwned!
      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
```



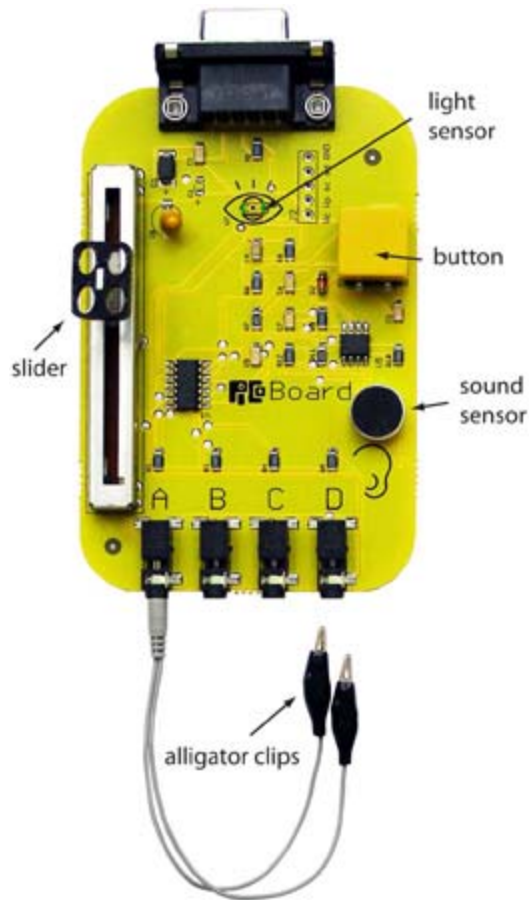
Events

Marco.sb



Sensors

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



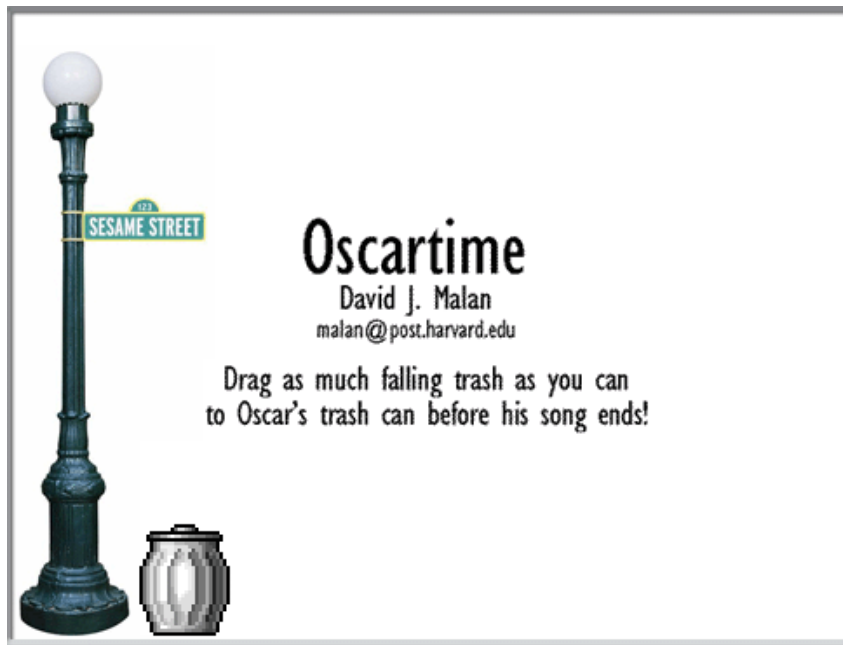
Oscartime

Oscartime.sb



Oscartime

Displaying the Instructions



Oscartime

Making Trash Fall



```
when green flag clicked
hide
point in direction 180
go to x: pick random -100 to 220 y: 180
wait 4 secs
show
go to front
forever
  if mouse down? and touching mouse-pointer? and my_click = 0 and good_click = 0
    set good_click to 1
    set my_click to 1
    broadcast trash_click
  if my_click = 0
    move 1 steps
  if playing = 1 and distance to Oscar < 20 and not mouse down?
    hide
    broadcast scored
    wait 2.25 secs
    go to x: pick random -100 to 220 y: 180
    show
```



Oscartime

Implementing Dragging



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

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
```

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 1.5 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
```

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
        set costume to oscar2
      else
        set costume to oscar1
```

Scratch Meets C



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

Statements

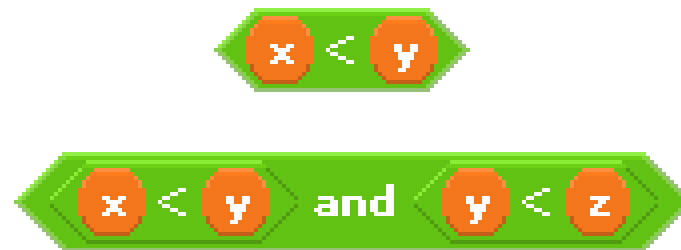
Scratch v. C



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

Boolean Expressions

Scratch v. C

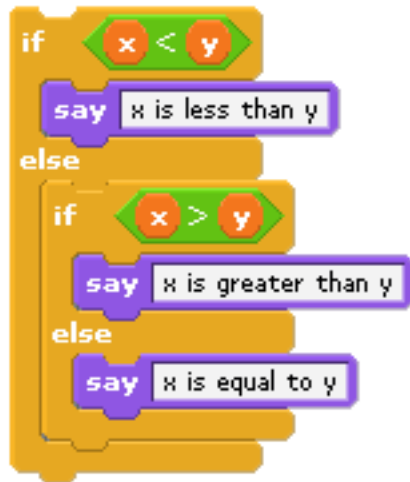


`(x < y)`

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

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");
}
```


Loops

Scratch v. C



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



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

Variables

Scratch v. C



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

Arrays

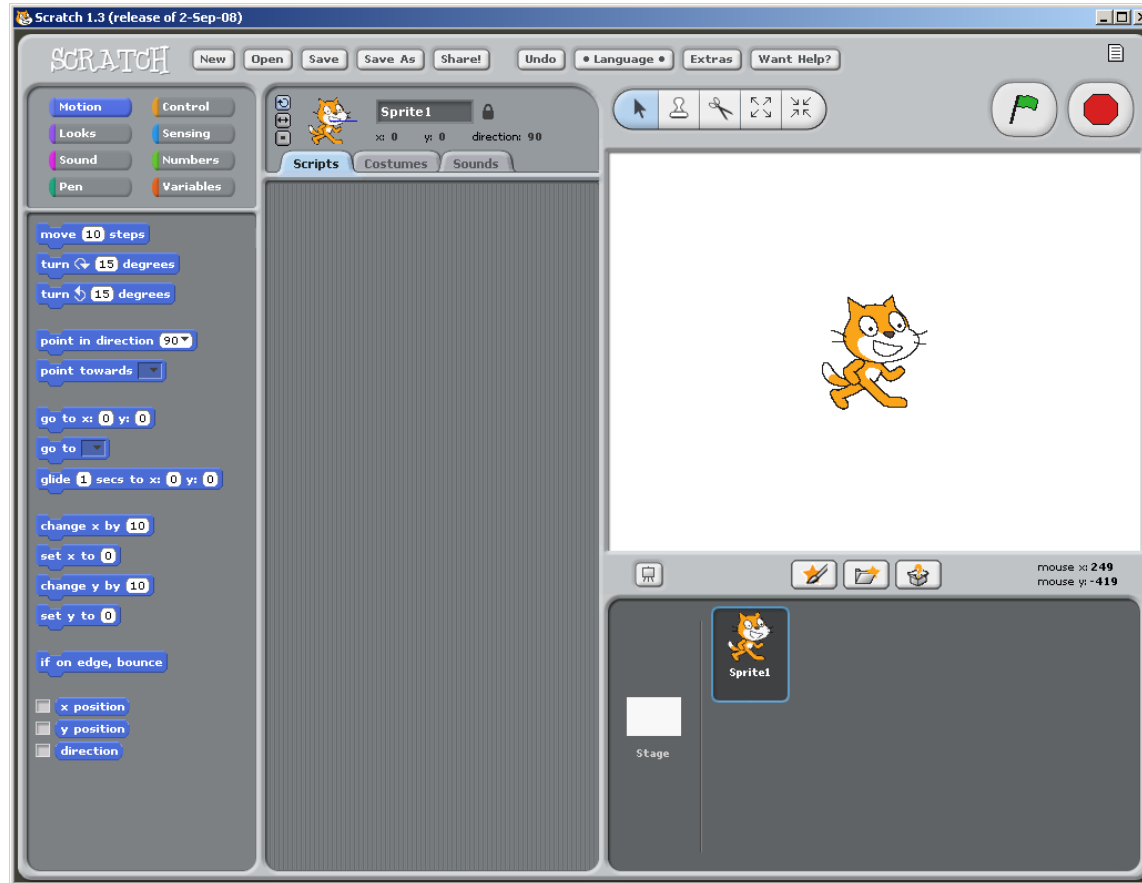
Scratch v. C



A Scratch code block with a blue tab labeled 'add', a text input field containing 'Orange', and a dropdown menu labeled 'inventory' with a small downward arrow.

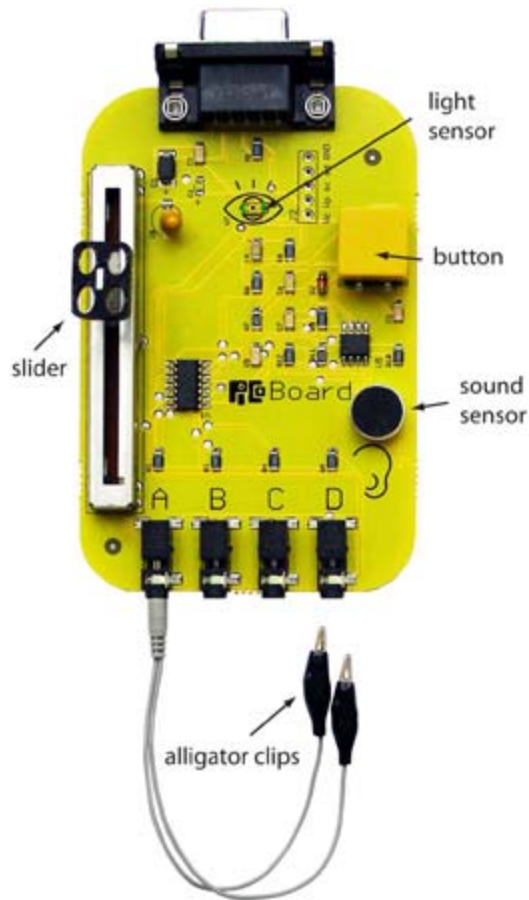
```
char *inventory[SIZE];  
inventory[i] = "Orange";
```

Problem Set 0



Problem Set 0

HACKER EDITION





Computer Science 50

Introduction to Computer Science I

Harvard College

Week 0

David J. Malan
malan@post.harvard.edu