Computer Science 50

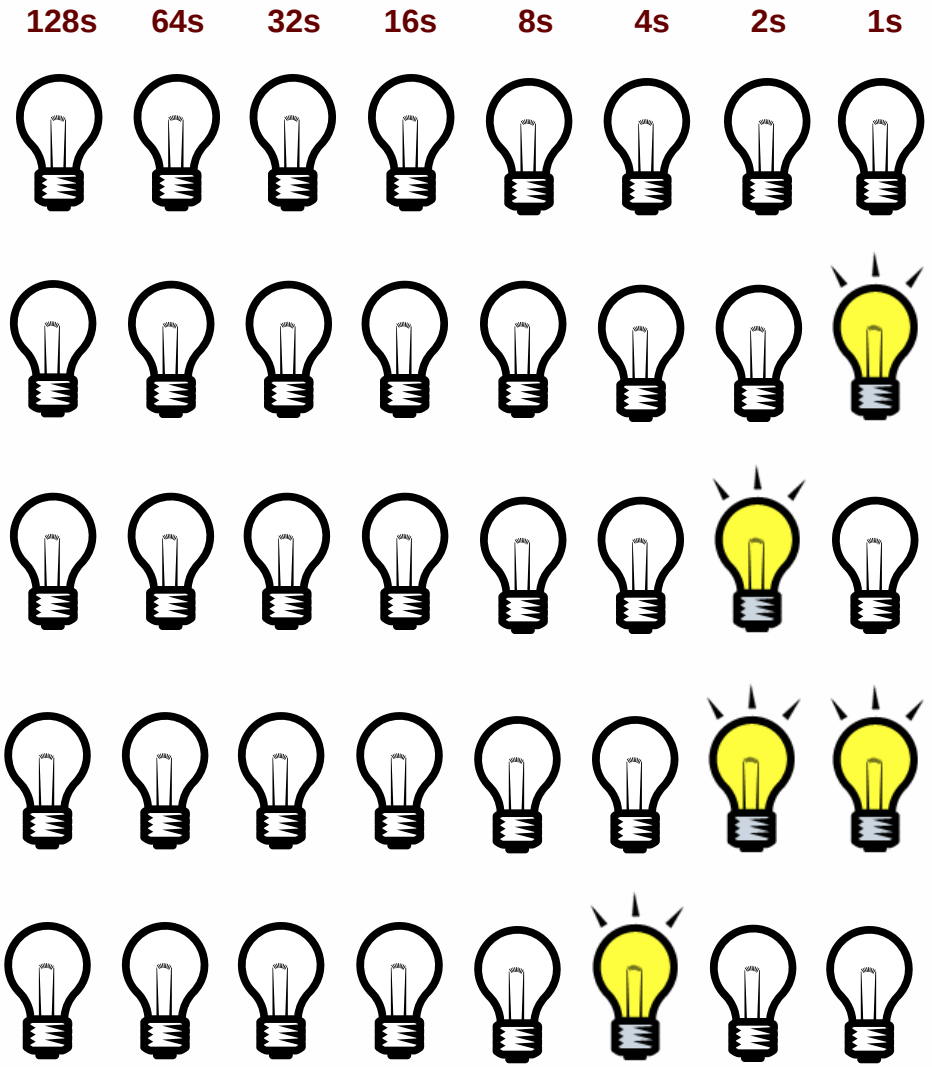
Introduction to Computer Science I

Harvard College

Week 0

David J. Malan
malan@post.harvard.edu

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

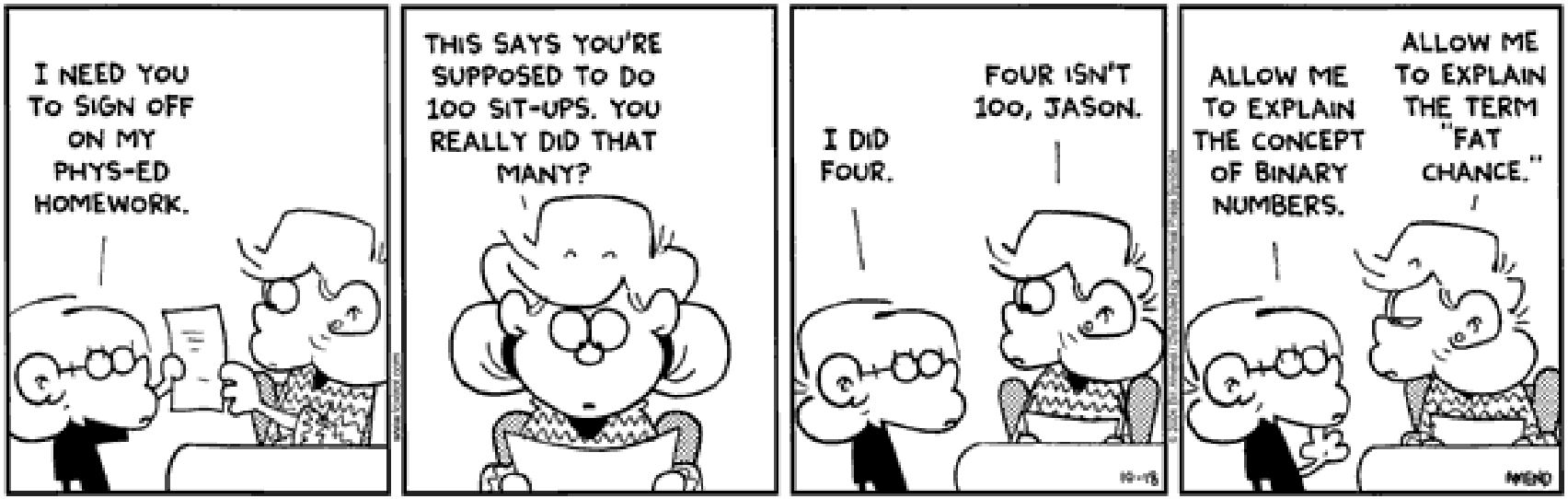
Cambridge in Binary

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	&#127;	DEL

Source: www.asciitable.com

Welcome to CS 50

Introduction to the intellectual enterprises of computer science. Algorithms: their design, specification, and analysis. Software development: problem decomposition, abstraction, data structures, implementation, debugging, testing. Architecture of computers: low-level data representation and instruction processing. Computer systems: programming languages, compilers, operating systems. Computers in the real world: networks, security and cryptography, artificial intelligence, social issues. Assignments include extensive programming in the C language and PHP.

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

Expectations

- Attend all lectures and sections
- Complete eight problem sets
- Take three quizzes
- Produce a final project

Grades*

- Problem Sets (best 7 out of 8) 65%
- Quizzes (best 2 out of 3) 20%
- Final Project 15%

* You may take the course pass/fail.

Website

www.fas.harvard.edu/~cs50/

(cs50.org)

Email

cs50@fas.harvard.edu

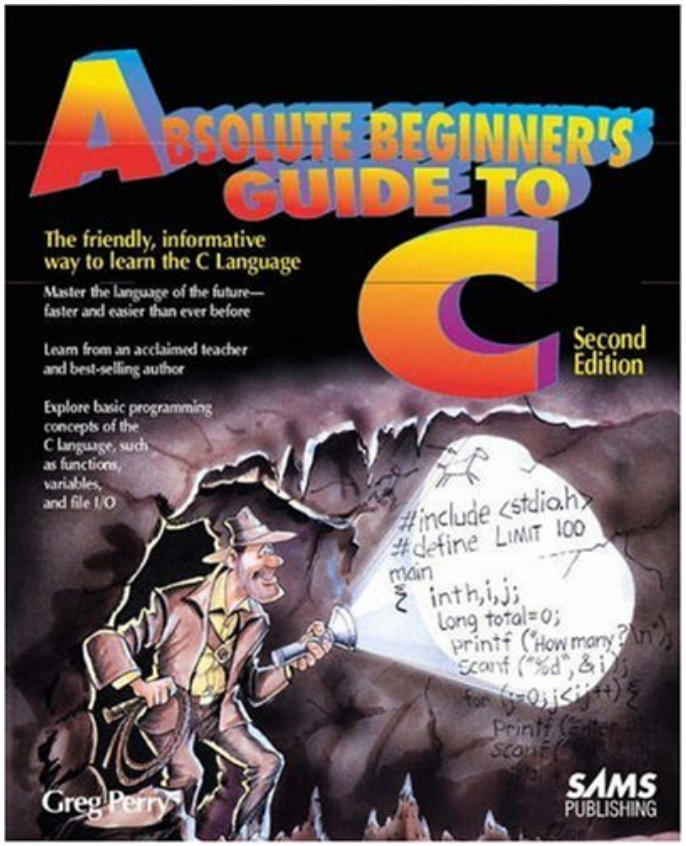
Bulletin Board

Forums

MAIN THEME	TOPICS	POSTS
Announcements	1	1
Final Project Ideas	1	1
Other	1	1
Problem Sets	1	1
Technical Support	1	1

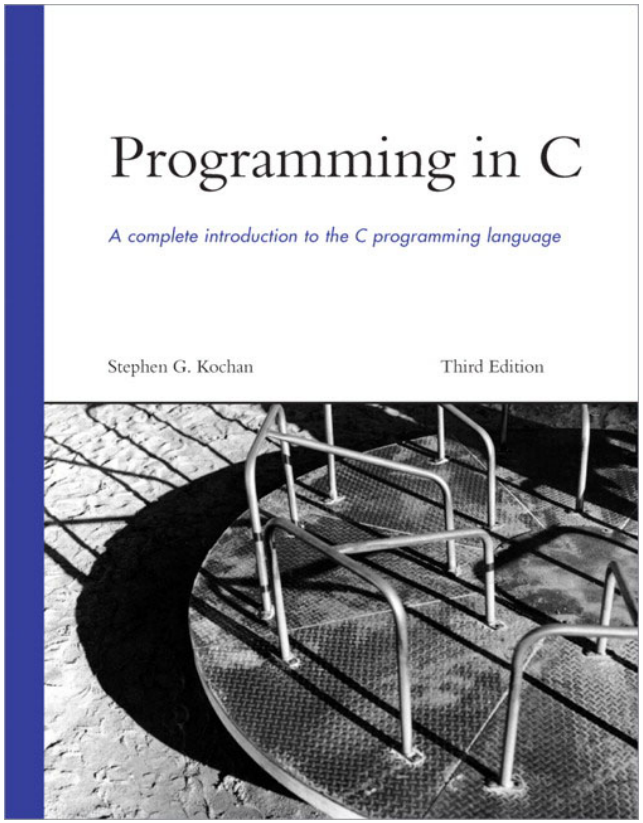
Books

For Those Less Comfortable



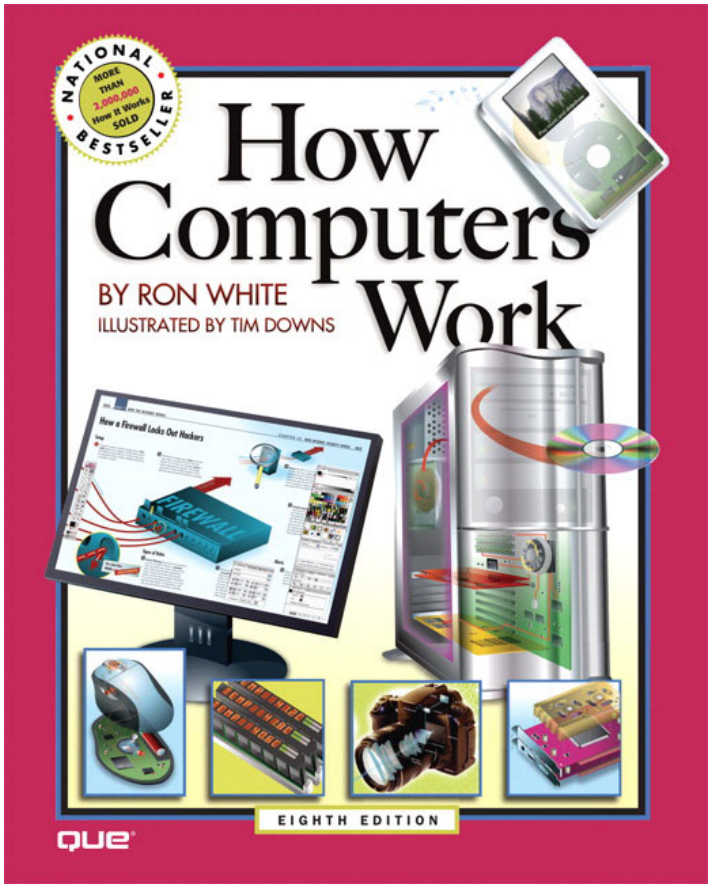
Books

For Those More Comfortable



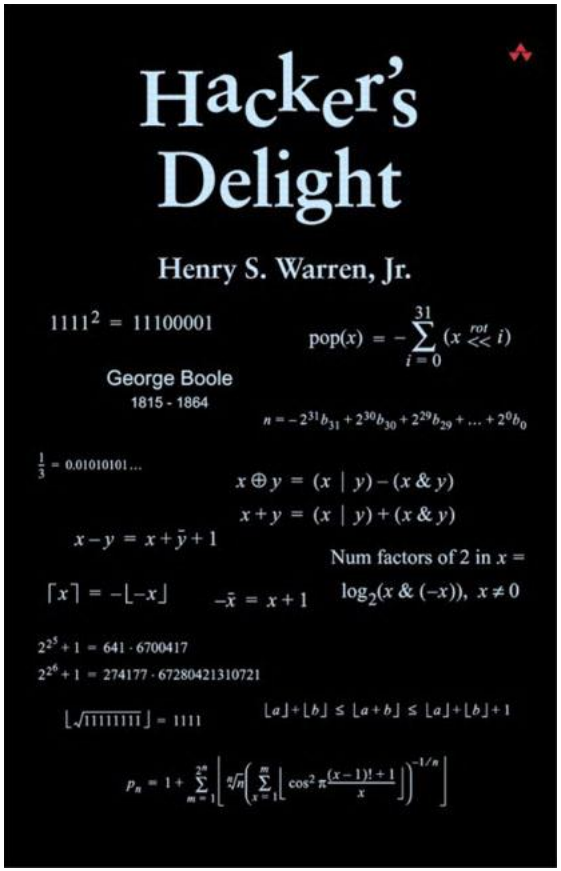
Books

For Everyone



Books

For Aspiring Hackers



Sections





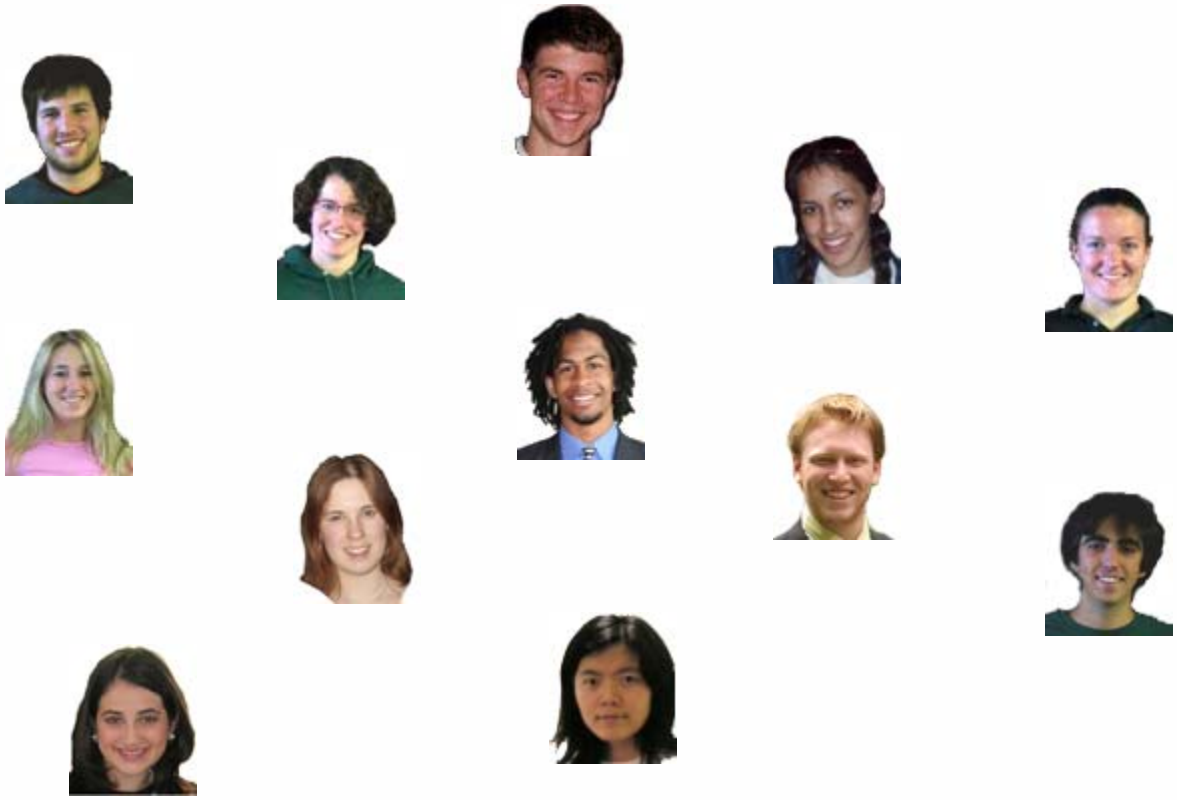






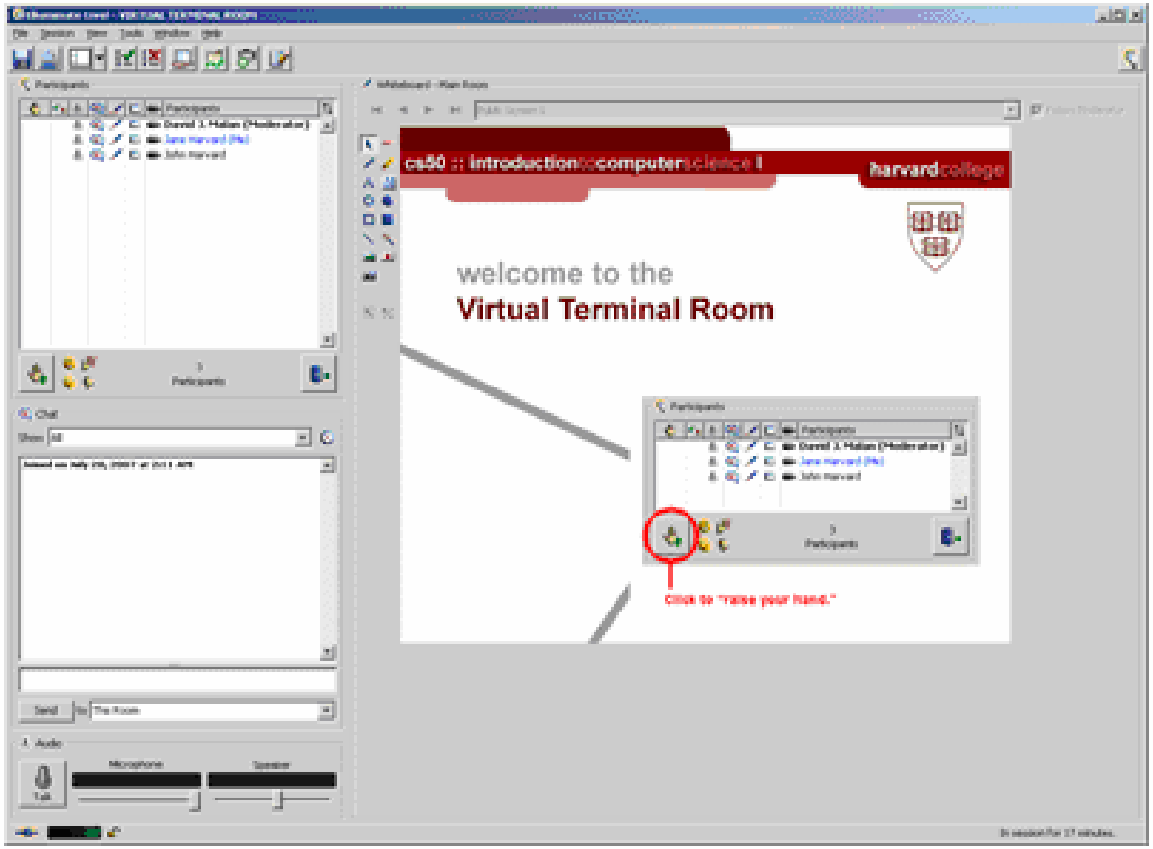

Head TFs

Supersections



Office Hours

Virtual Office Hours



Lectures

Week 0

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

Lectures

Week 1

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

```
#include <stdio.h>

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

Lectures

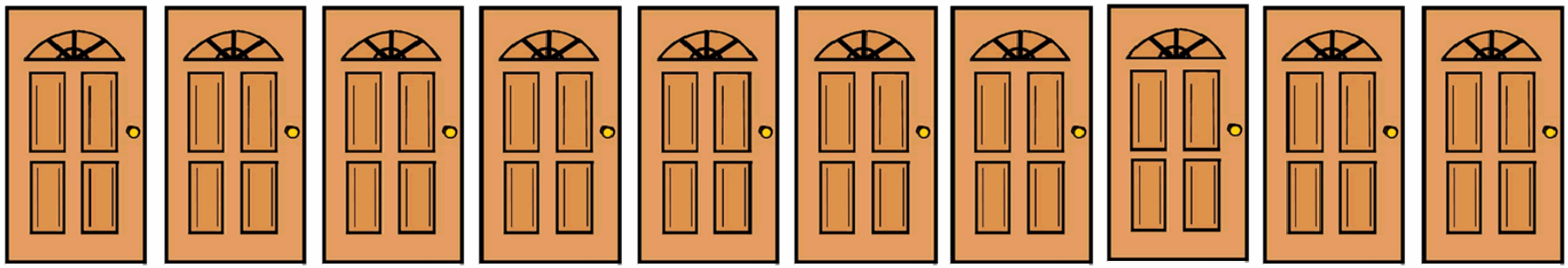
Week 2

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. Selection sort. Bubble sort. Insertion sort. Pseudorandomness.



Lectures

Week 4

Debugging software. Designing software.

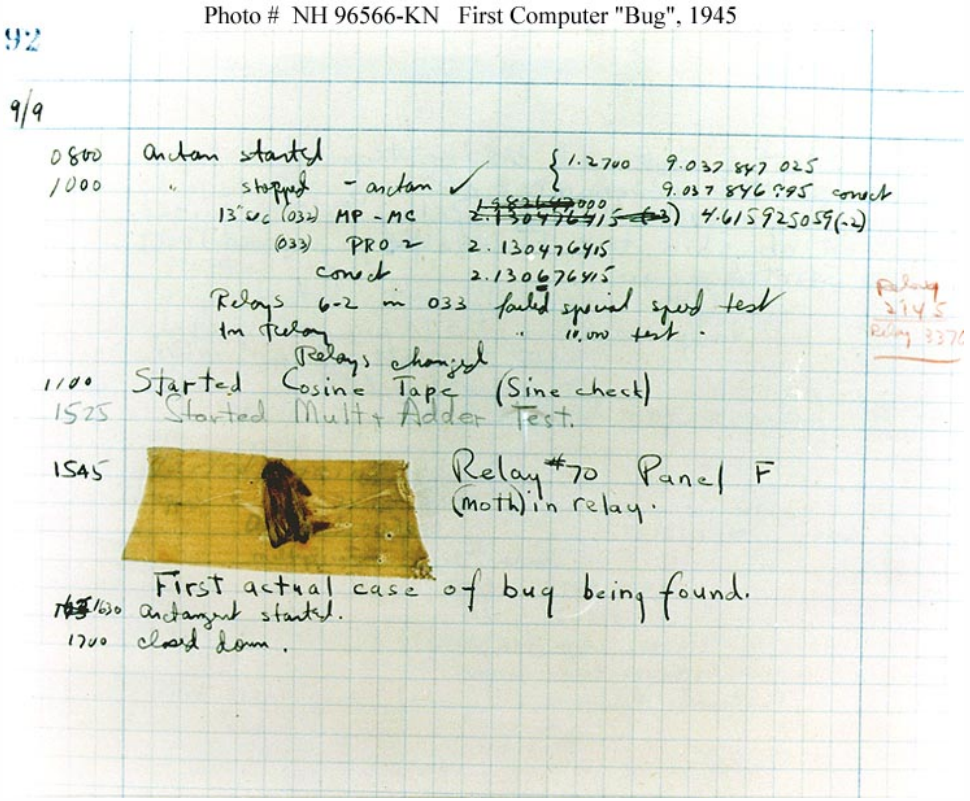
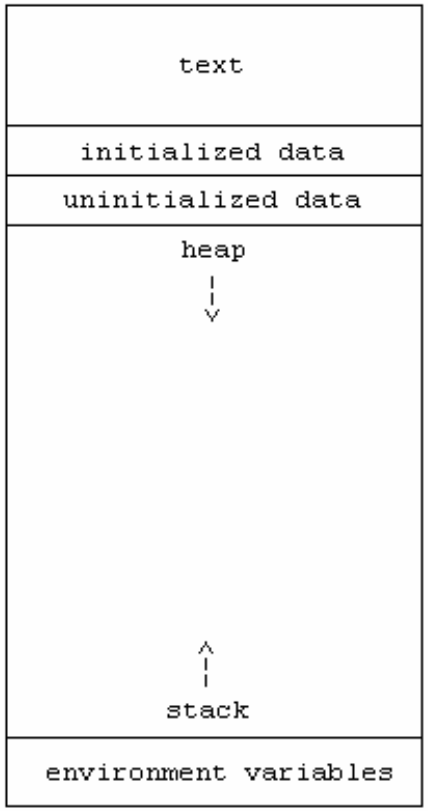


Image from http://www.history.navy.mil/.

Lectures

Week 5

Mergesort. Quicksort. Structures. Dynamic memory allocation. Pointers. Heap. Digital forensics. File I/O.



Lectures

Week 6

Singly linked lists. Doubly linked lists. Binary search trees. Heaps. Heapsort. Hash tables. Tries.

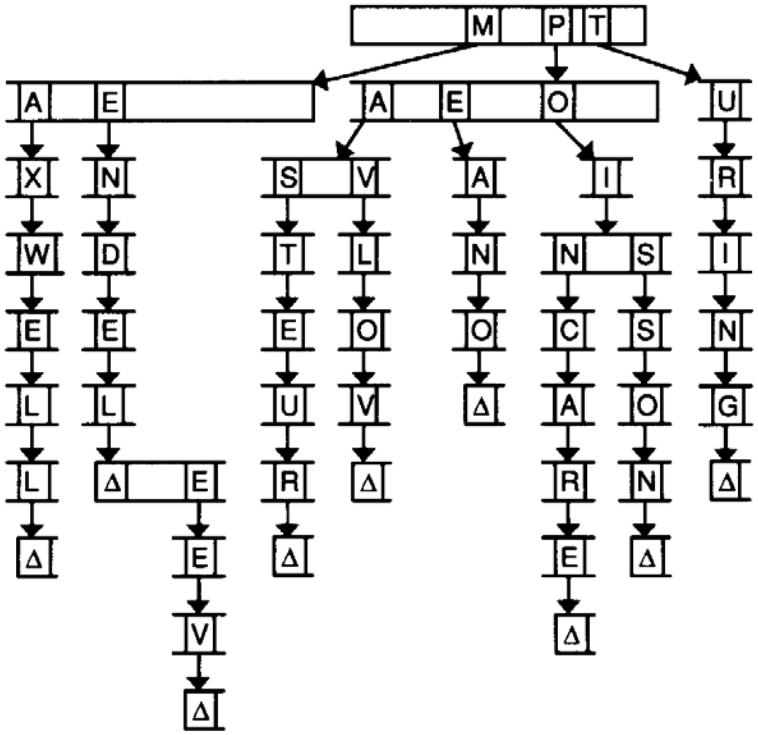


Image from *Data Structures & Their Algorithms.*

Lectures

Week 7

Huffman coding. Bitwise operators.

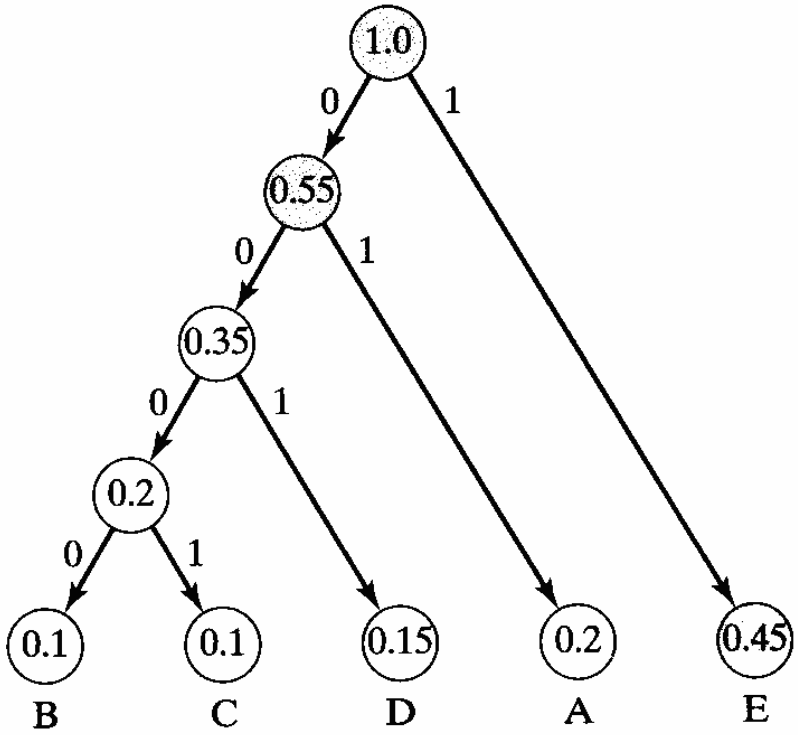


Image from *C++: An Introduction to Data Structures.*

Lectures

Week 8

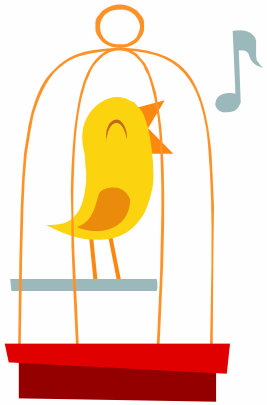
Preprocessing. Compiling. Assembling. Linking. CPUs. Ant-8.

```
# Dan Ellard -- 01/19/2000
# hello.asm-- An Ant-8 "Hello World" program.
# Registers used:
# r2  - holds the address of the string
# r3  - holds the address of the end of the loop
# r4  - holds the next character to be printed.

        lc      r2, $str_data   # load the address of the string into r2
        lc      r3, $endloop    # load address of the end of loop.

loop:
        ld1     r4, r2, 0       # Get the first character from the string
        beq     r3, r4, r0      # If the char is zero, we're finished.
        out     r4, ASCII       # Otherwise, print the character.
        inc     r2, 1           # Increment r2 to point to the next char
        jmp     $loop           # and repeat the process...

endloop:
        hlt

_data_:                         # Data for the program begins here:

str_data:
        .byte   'H', 'e', 'l', 'l', 'o', ' '
        .byte   'W', 'o', 'r', 'l', 'd', '\n', 0

# end of hello.asm
```

Lectures

Week 9

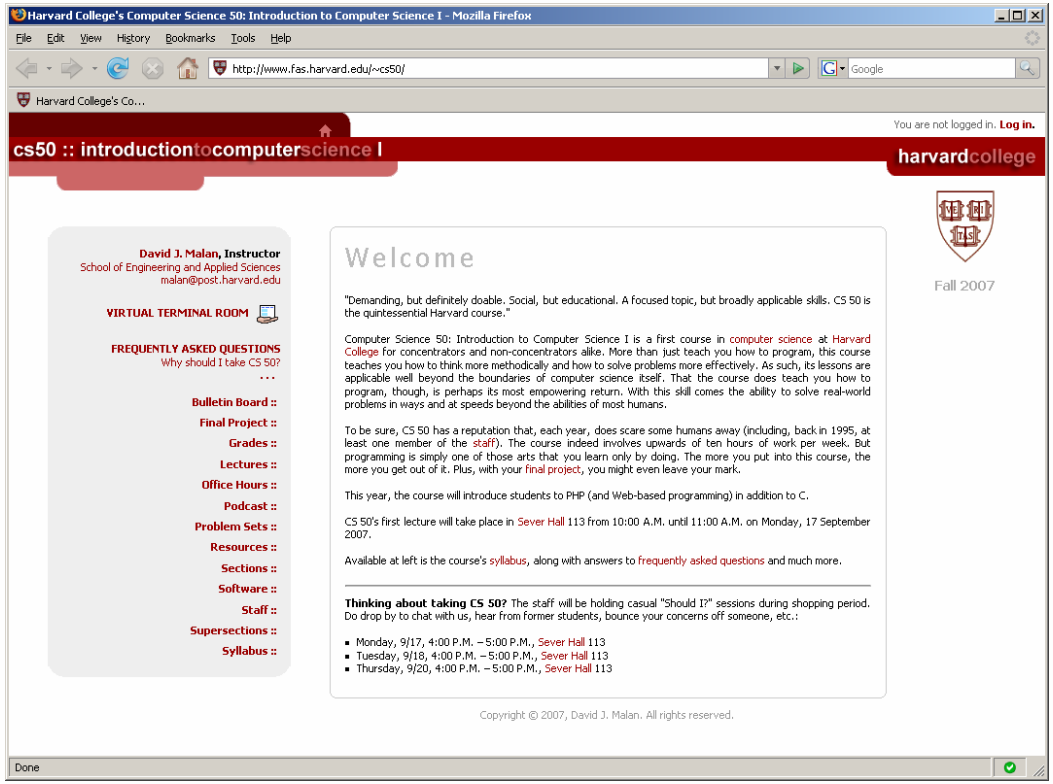
Secure coding.



Lectures

Week 10

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



Lectures

Week 11

To be announced.

Lectures

Week 12

Fun. Exciting conclusion.

Problem Sets*

- Problem Set 0: Scratch
- Problem Set 1: C
- Problem Set 2: Crypto
- Problem Set 3: The Game of 15
- Problem Set 4: Forensics
- Problem Set 5: Mispellings**
- Problem Set 6: Huff’n Puff
- Problem Set 7: XHTML + PHP + SQL

* Hacker Editions too.

** Yes, we know.

Final Project*

- Make something useful.
- Solve an actual problem.
- Somehow impact campus.

* Strive to make something that outlives this course.

Should I (take CS 50)?

- Monday, 9/17, 4p – 5p, Sever 113
- Tuesday, 9/18, 4p – 5p, Sever 113
- Thursday, 9/20, 4p – 5p, Sever 113

Programming

Algorithms

1) Stand up.
2) Assign yourself the number 1.
3) Find someone else that is standing up.
   (If no one is standing, remain standing until I call on you.)
4) Add your number to that person's number;
   the total is your new number.
5) One of you should then sit down.
6) If you're still standing, go back to step 3.

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
```

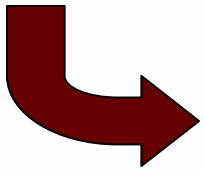
Hello, C!

hello.c

```
#include <stdio.h>

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

Hello, C!

```
#include <stdio.h>

int
main(int argc, char * argv[])
{
    printf("hello, 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
[...]
```

Hello, Scratch!

Hello1.sb

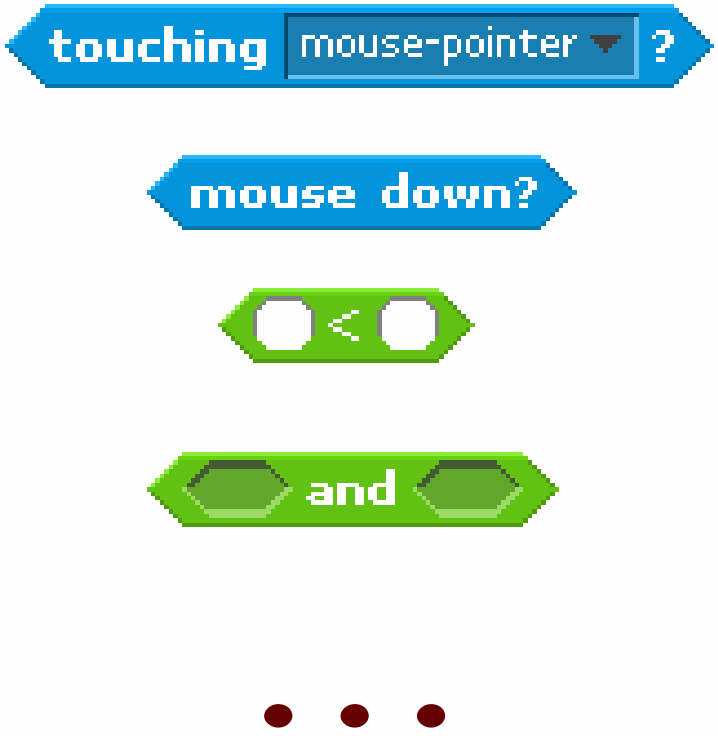
Statements

• • •

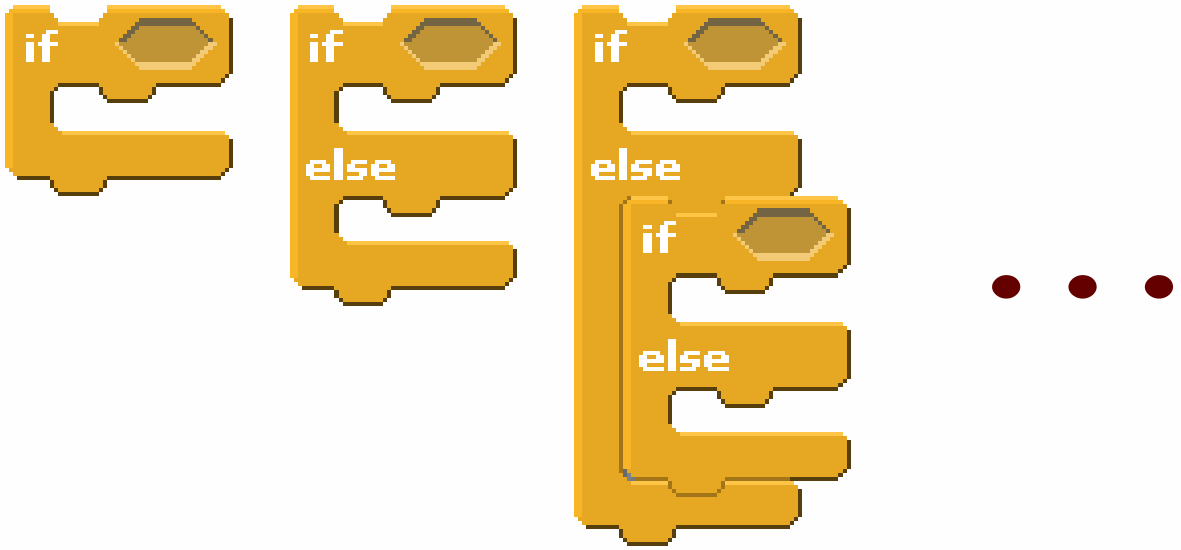
Statements

Hello{2,3}.sb

Boolean Expressions



Conditions



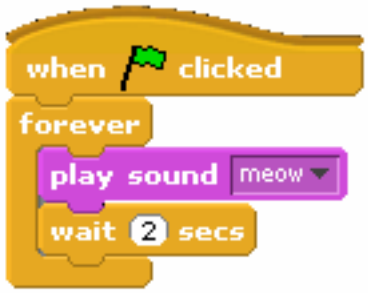
Conditions

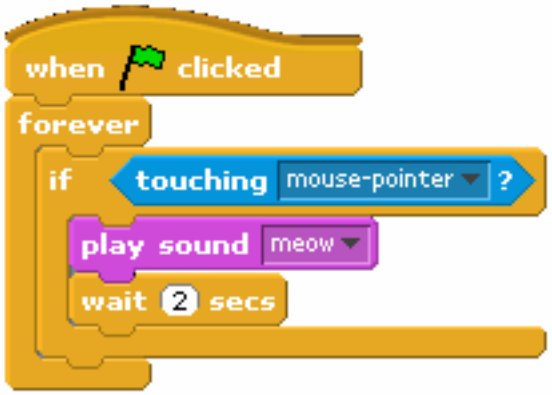
Hello{4,5}.sb

Loops

Loops

Hello{6,7,8}.sb





Variables

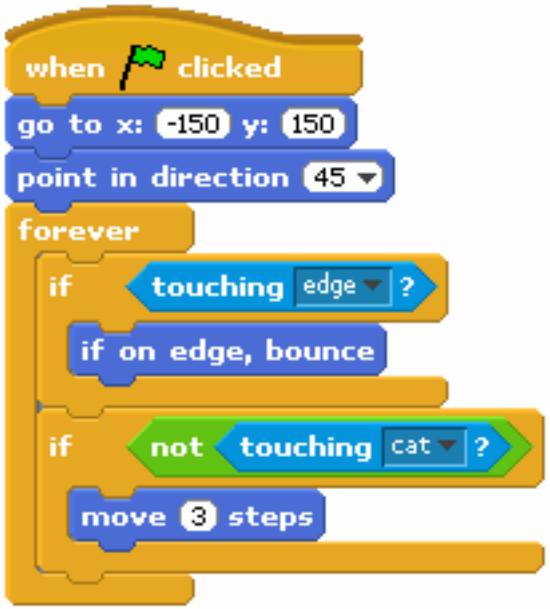
{Count,Hello9}.sb

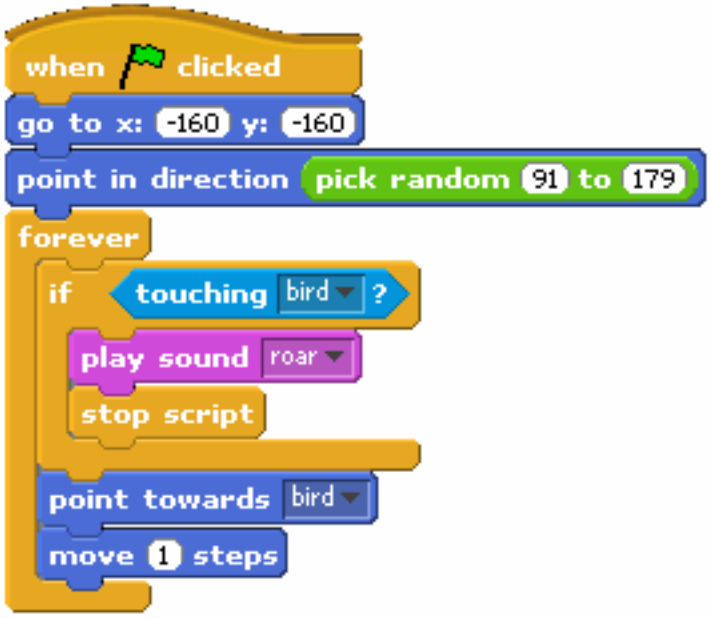
Threads

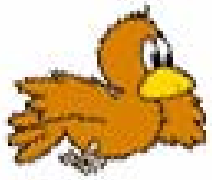
Move1.sb

Threads

Move2.sb

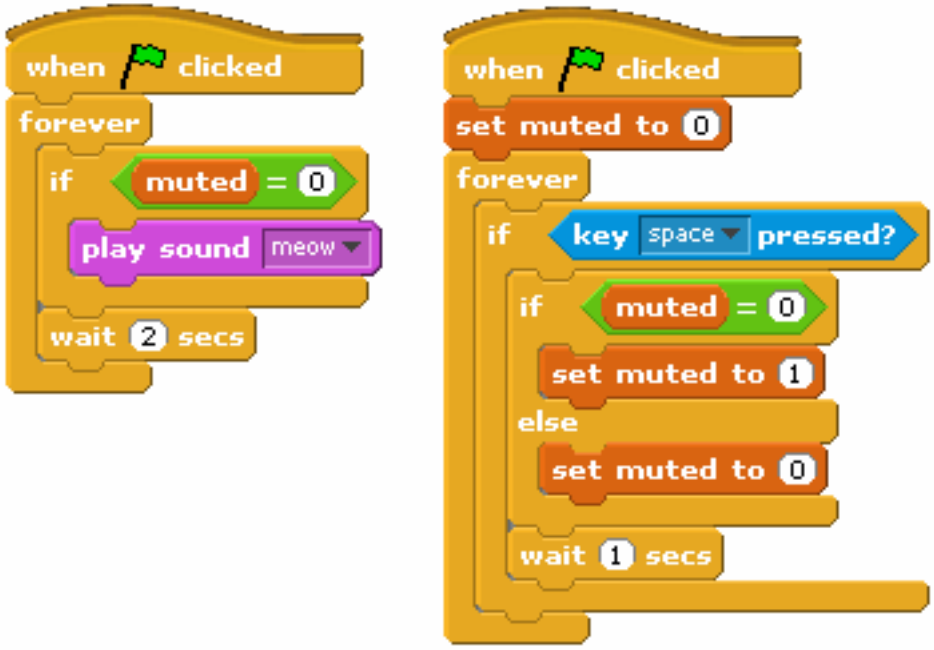






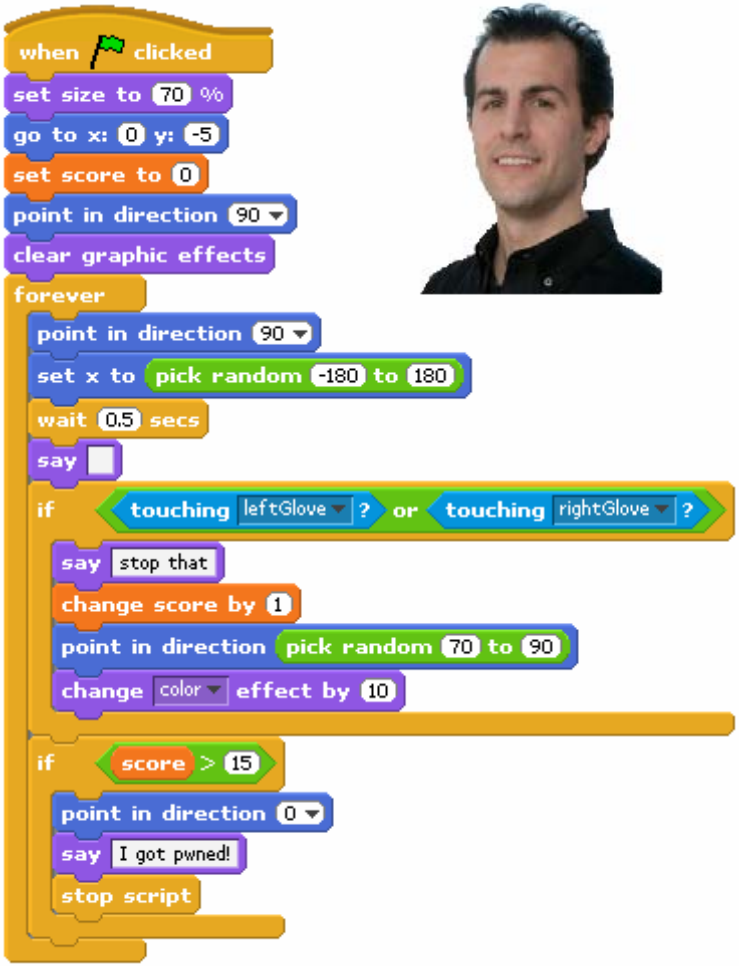
Threads

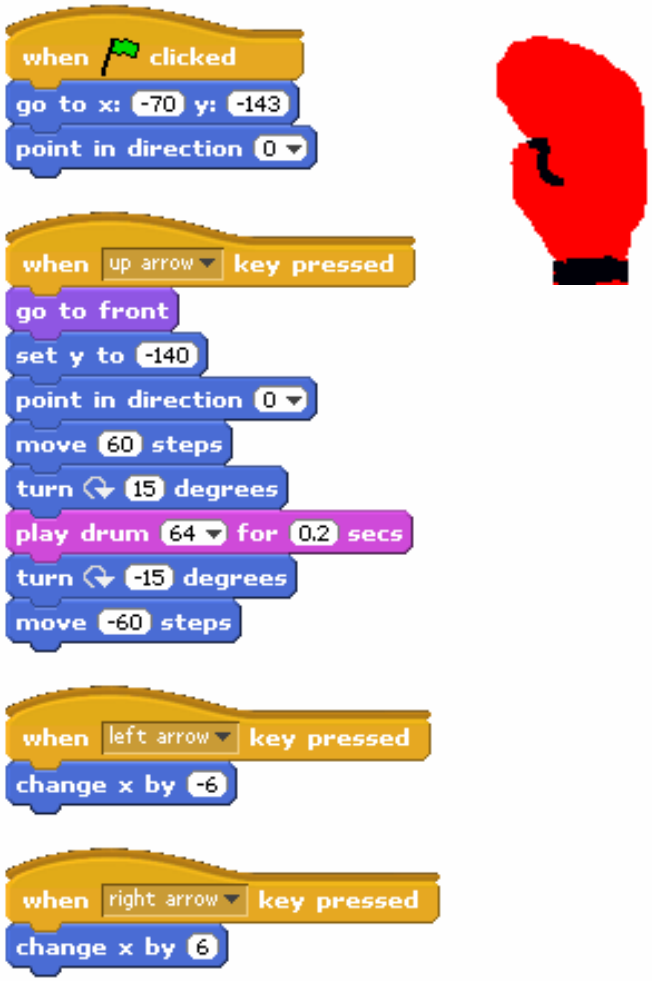
Hello10.sb



Threads

David.sb





Events

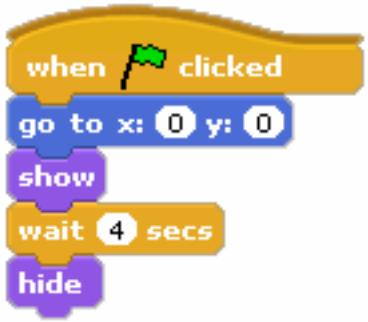
Marco.sb

Oscartime

Oscartime.sb

Oscartime

Displaying the Instructions



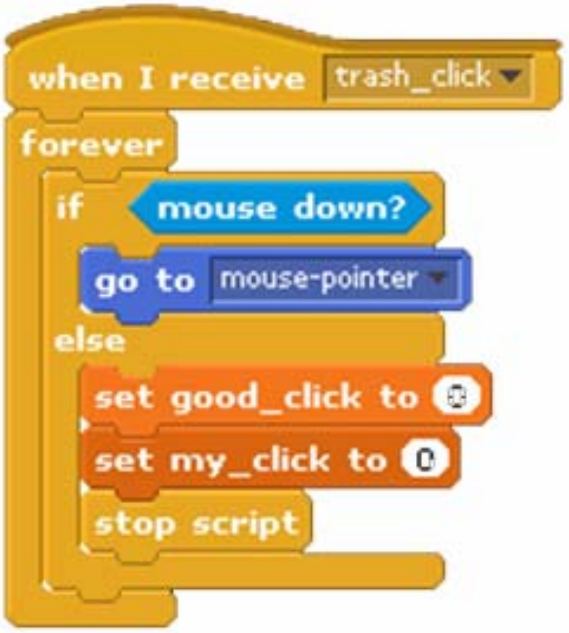
Oscartime

Making Trash Fall


```
when [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
```

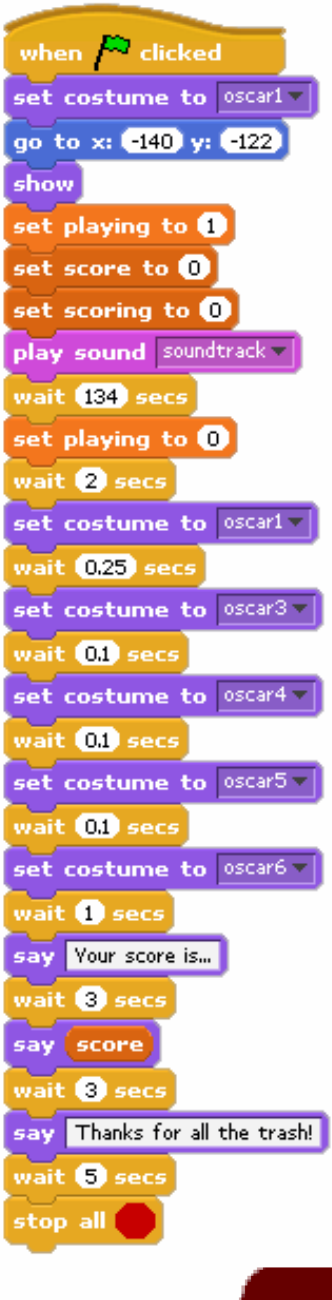
Oscartime

Implementing Dragging



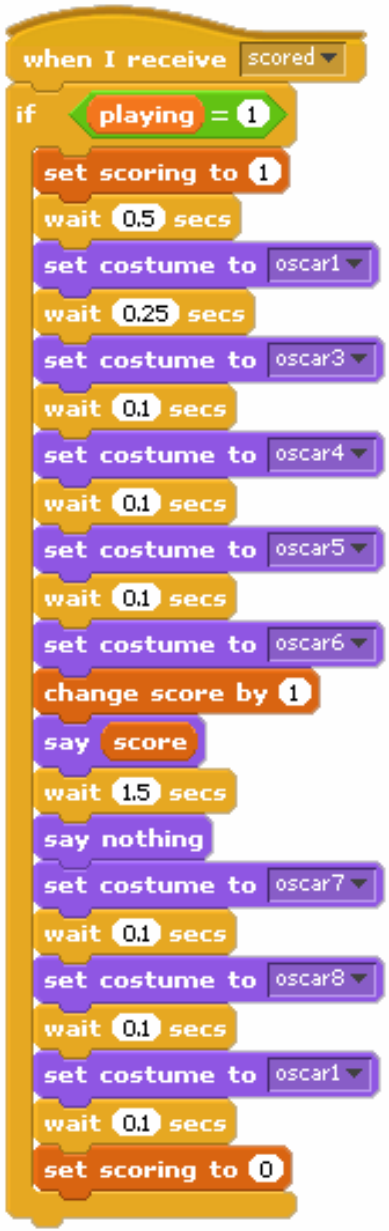
Oscartime

Imposing a Time Limit



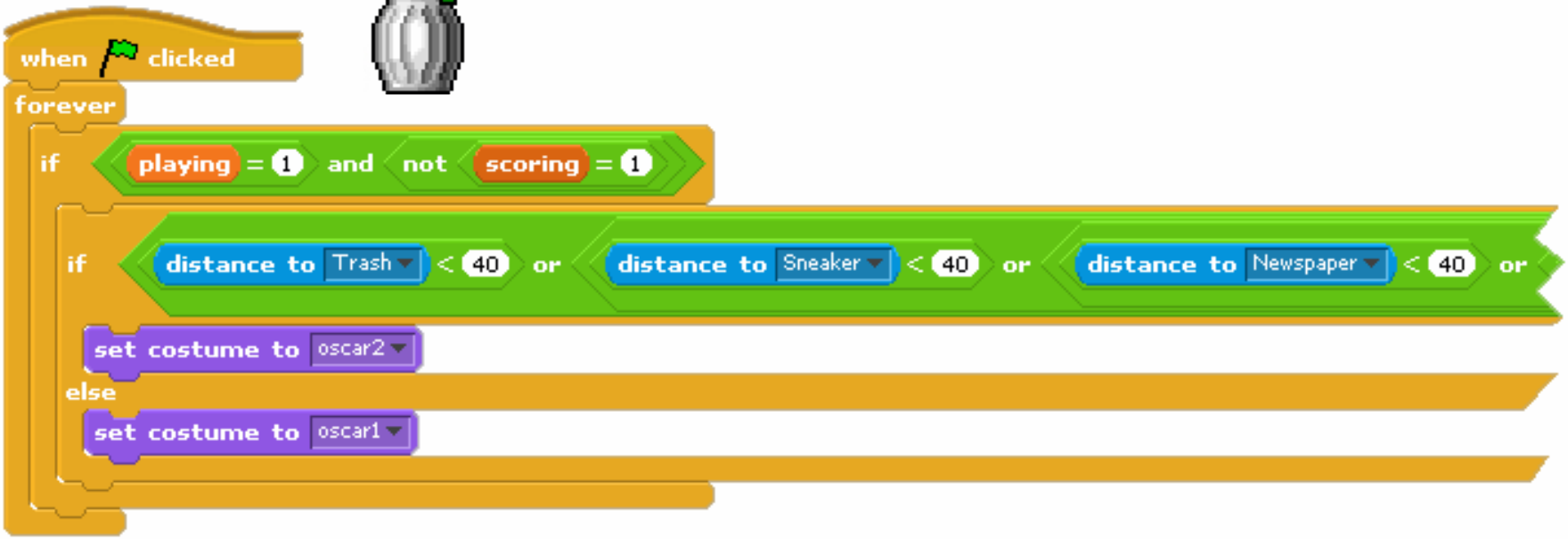
Oscartime

Keeping Score



Oscartime

Raising Oscar's Lid



Scratch Meets C


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

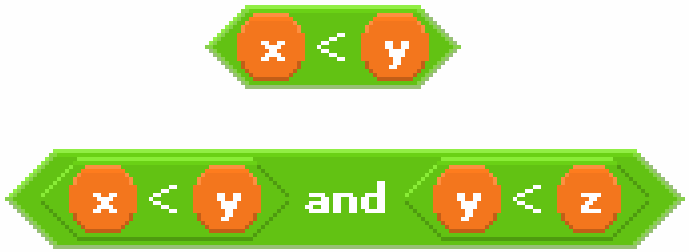
Statements

Scratch v. C


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

Boolean Expressions

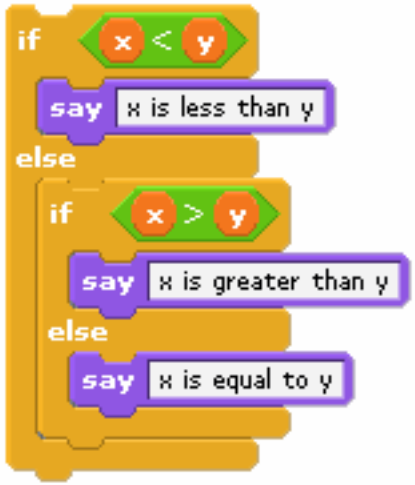
Scratch v. C




```
(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

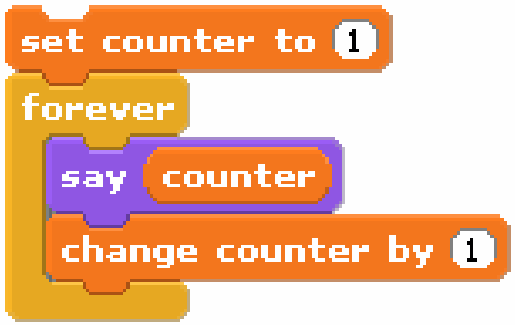
Scratch v. C


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


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

Variables

Scratch v. C




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

Computer Science 50

Introduction to Computer Science I

Harvard College

Week 0

David J. Malan
malan@post.harvard.edu