UC Berkeley EECS
Lecturer
Gerald Friedland

CS10
The Beauty and Joy of Computing

Lecture #17
Higher Order Functions

2014-11-03

SELF-DRIVING CARS

PRO

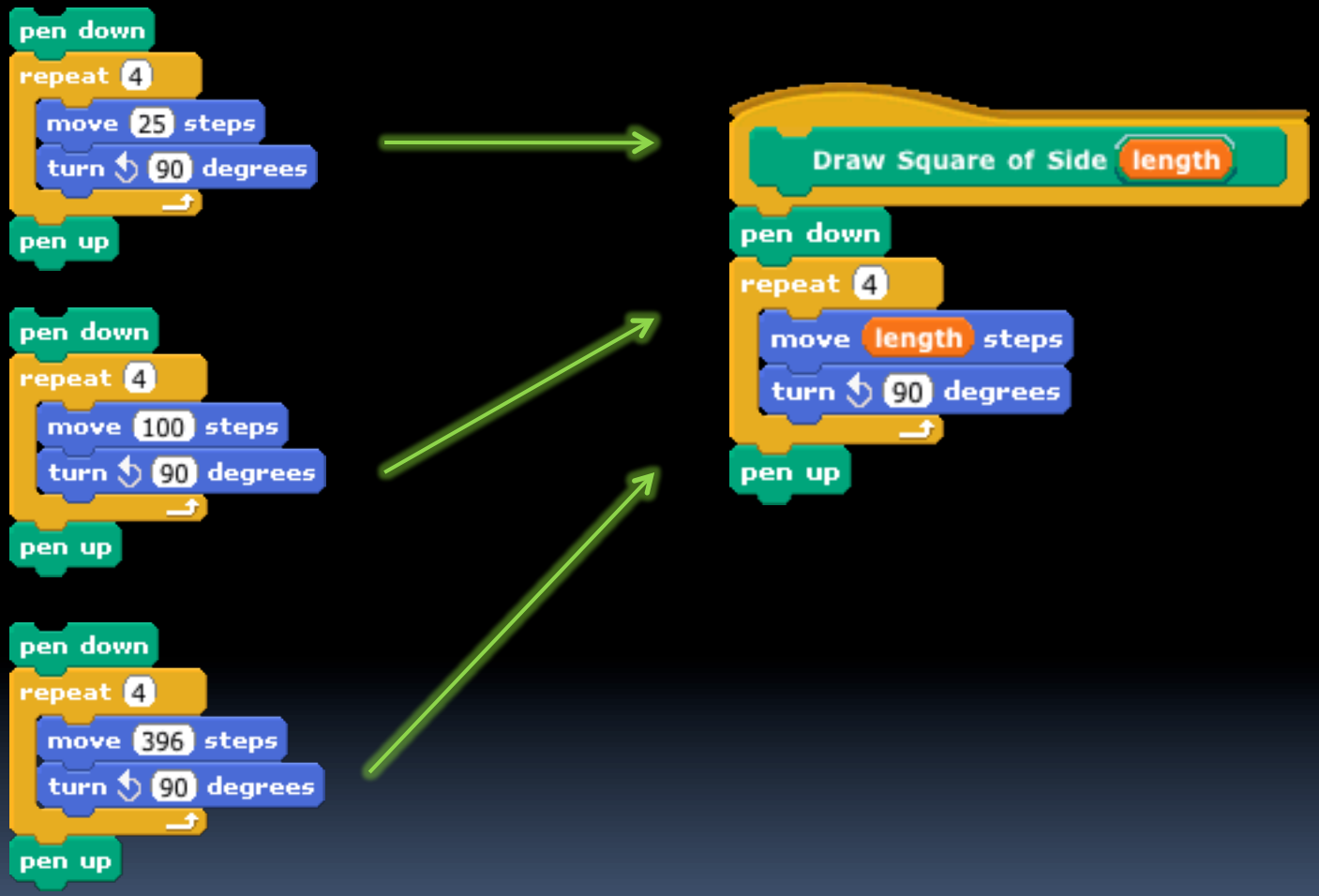
- Fewer accidents – 90% of accidents caused by human error
- Efficient travel since can create convoys
- Huge efficiency gains if you can work + drive

CON

- Who gets sued when there's an accident?
- Handing control back to driver takes ~5 sec
- Very expensive
- Could be dangerous if they can't handle case

www.technologyreview.com/featuredstory/520431/driverless-cars-are-further-away-than-you-think/

Why use functions? (review)



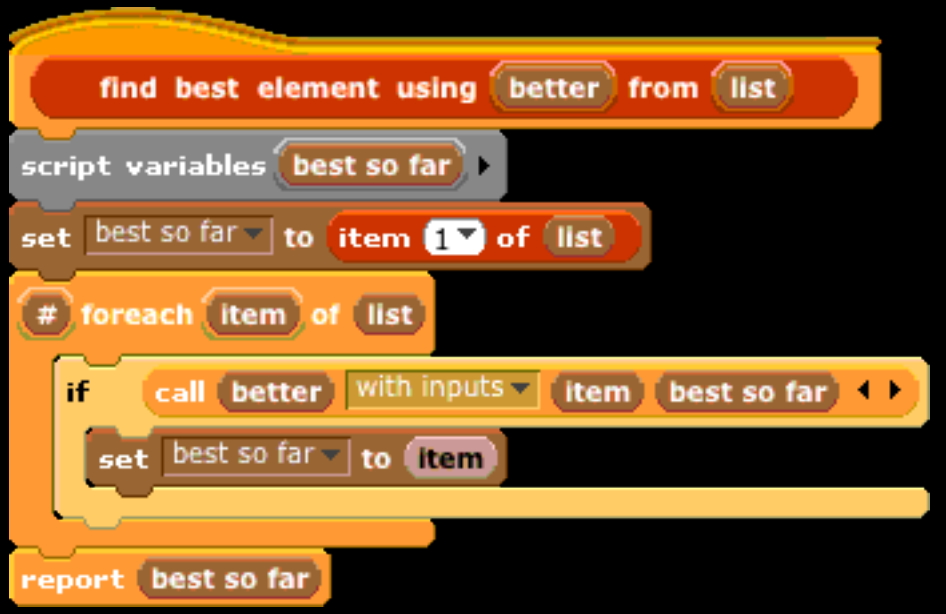
The power of **Abstraction**!

Peer Instruction

I understand functions.

a) Strongly disagree
b) Disagree
c) Neutral
d) Agree
e) Strongly agree

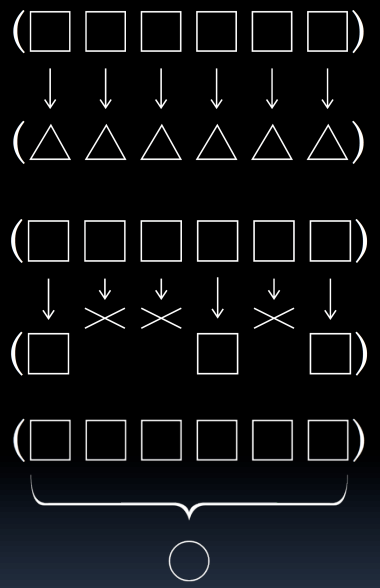
But how general can we be?

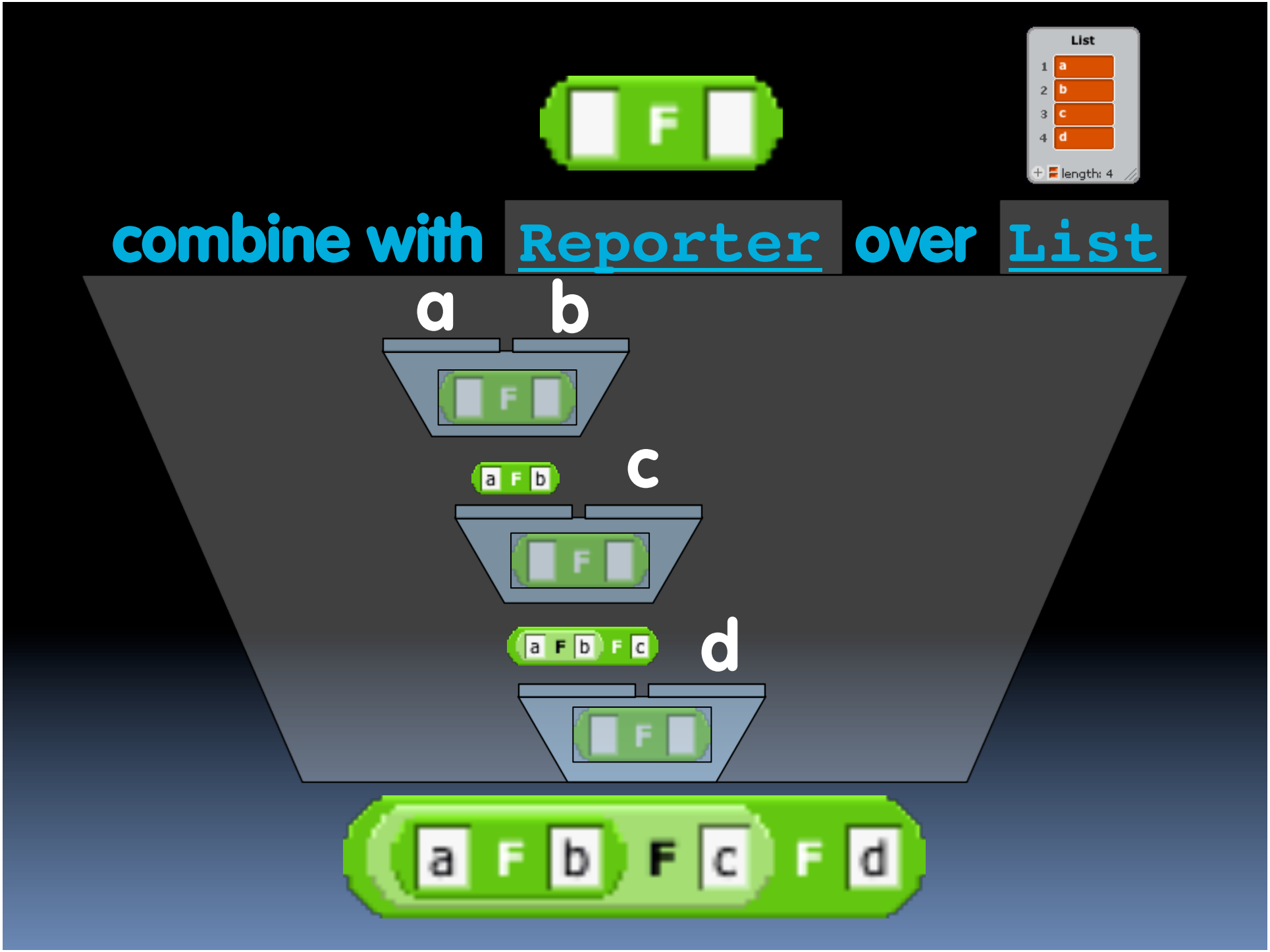


The power of even more Abstraction!

Today

- Functions as Data
- Higher-Order Functions
- Useful HOFs (you can build your own!)
  - map `Reporter` over `List`
    - Report a new list, every element `E` of `List` becoming `Reporter(E)`
  - keep items such that `Predicate` from `List`
    - Report a new list, keeping only elements `E` of `List` if `Predicate(E)`
  - combine with `Reporter` over `List`
    - Combine all the elements of `List` with `Reporter(E)`
    - This is also known as "reduce"
- Acronym example
  - keep → map → combine




F
List
1 a
2 b
3 c
4 d
length: 4
combine with Reporter over List
a
b
F
a F b
c
F
a F b F c
d
F
a F b F c F d

Peer Instruction

I understand higher-order functions.

a) Strongly disagree
b) Disagree
c) Neutral
d) Agree
e) Strongly agree

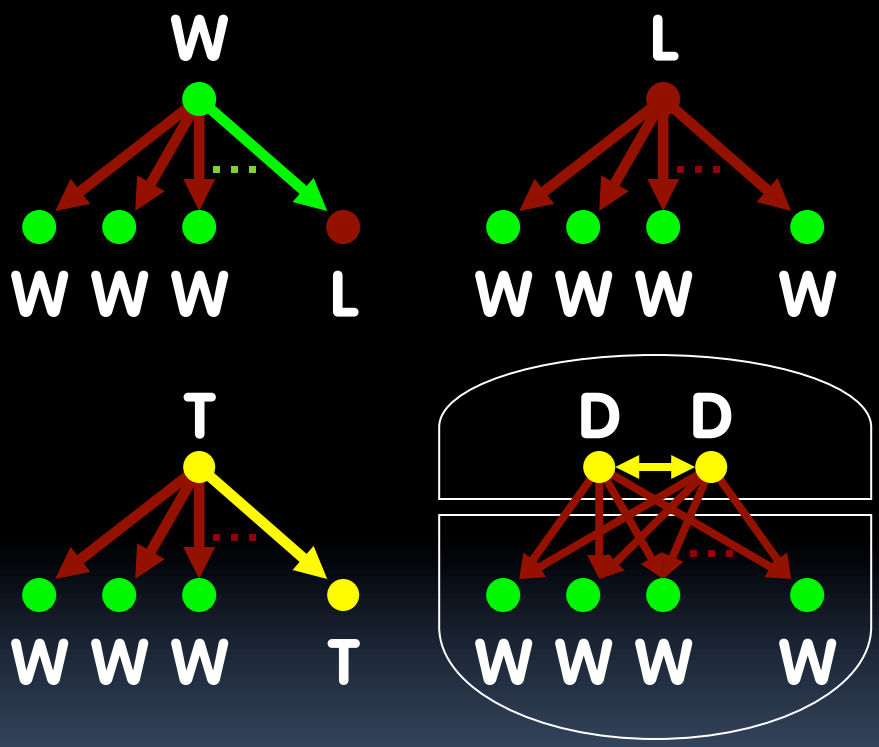
Friedland

Let's Play Board Games…

- No chance, such as dice or shuffled cards
- Both players have **complete information**
  - No hidden information, as in Stratego & Magic
- Two players (Left & Right) usually alternate moves
  - Repeat & skip moves ok
  - Simultaneous moves not ok
- The game can end in a pattern, capture, by the absence of moves, or …

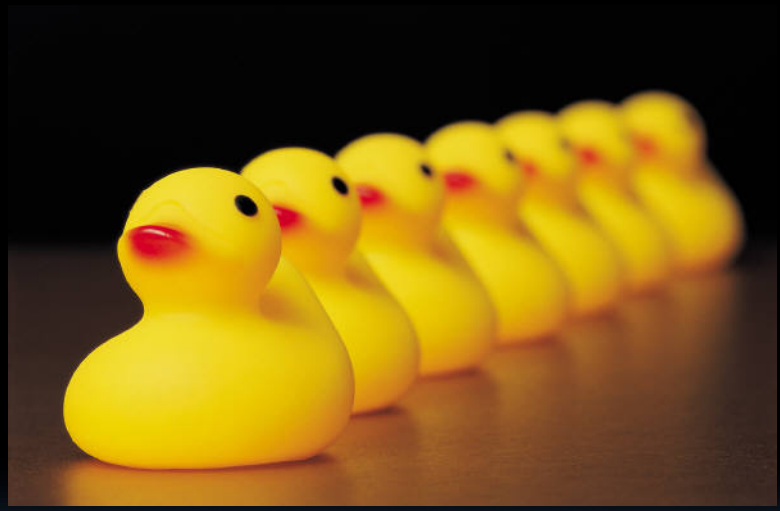
A Strong Solution visits every position

- **For every position**
  - Assuming alternating play
  - Value …
    (for player whose turn it is)
    - Winning (losing child)
    - Losing (All children winning)
    - Tieing (! losing child, but tieing child)
    - Drawing (can't force a win or be forced to lose)
  - Remoteness
    - How long before game ends?



Strong Solving Example: 1,2,…,10

- **Rules (on your turn):**
  - Running total = 0
- **Rules (on your turn):**
  - Add 1 or 2 to running total
- **Goal**
  - Be the FIRST to get to 10
- **Example**
  - Ana: "2 to make it 2"
  - Bob: "1 to make it 3"
  - Ana: "2 to make it 5"
  - Bob: "2 to make it 7" → photo
  - Ana: "1 to make it 8"
  - Bob: "2 to make it 10" I WIN!



7 ducks (out of 10)

Let's write code to determine value!

- 0 = Win
- 1 = Lose
- 2 = Win
- 3 = Win
- 4 = Lose
- 5 = Win
- 6 = Win
- 7 = Lose
- 8 = Win
- 9 = Win
- 10 = Lose

- P = Position
- M = Move
- We only need 3 blocks to define a game
  - Do Move M on Position P
    - → a new Position
  - Generate Moves from Position P
    - → list of Moves
  - Primitive Value of Position P
    - → {win, lose, tie, undecided}
- Let's write **Value of Position P**

Friedland

Answer

A Chess Engine…

```
/************************************************************************\
| Toledo Nanochess (c) Copyright 2009 Oscar Toledo G. All rights reserved  |
| 1257 non-blank characters. Evolution from my winning IOCCC 2005 entry.   |
| o Use D2D4 algebraic style for movements.  biyubi@gmail.com  Nov/20/2009 |
| o On promotion add a number for final piece (3=N, 4=B, 5=R, 6=Q)         |
| o Press Enter alone for computer to play.                                |
| o Full legal chess moves.                          http://www.nanochess.org |
| o Remove these comments to get 1326 bytes source code (*NIX end-of-line) |
\************************************************************************/
char*l="ustvrtsuqqqqqqqqqyyyyyyyyy}{|~z|{}"
"   76Lsabcddcba .pknbrq  PKNBRQ ?A6J57IKJT576,+-48HLSU";
#define F getchar()&z
#define v X(0,0,0,21,
#define Z while(
#define _ ;if(
#define P return--G,y^=8,
B,i,y,u,b,I[411],*G=I,x=10,z=15,M=1e4;X(w,c,h,e,S,s){int t,o,L,E,d,O=e,N=-M*M,K
=78-h<<x,p,*g,n,*m,A,q,r,C,J,a=y?-x:x;y^=8;G++;d=w||s&&s>=h&&v 0,0)>M;do{_ o=I[
p=0]){q=o&z^y _ q<7){A=q--&2?8:4;C=o-9&z?q["& .$  "]:42;do{r=I[p+=C[l]-64]_!w|p
==w){g=q|p+a-S?0:I+S _!r&(q|A<3||g)||(r+1&z^y)>9&&q|A>2){_ m=!(r-2&7))P G[1]=O,
K;J=n=o&z;E=I[p-a]&z;t=q|E-7?n:(n+=2,6^y);Z n<=t){L=r?l[r&7]*9-189-h-q:0 _ s)L
+=(1-q?l[p/x+5]-l[O/x+5]+l[p%x+6]*-~!q-l[O%x+6]+o/16*8:!!m*9)+(q?0:!(I[p-1]^n)+
!(I[p+1]^n)+l[n&7]*9-386+!!g*99+(A<2))+!(E^y^9)_ s>h||1<s&s==h&&L>z|d){p[I]=n,O
[I]=m?*g=*m,*m=0:g?*g=0:0;L-=X(s>h|d?0:p,L-N,h+1,G[1],J=q|A>1?0:p,s)_!(h||s-1|B
-O|i-n|p-b|L<-M))P y^=8,u=J;J=q-1|A<7||m||!s|d|r|o<z||v 0,0)>M;O[I]=o;p[I]=r;m?
*m=*g,*g=0:g?*g=9^y:0;}_ L>N){*G=O _ s>1){_ h&&c-L<0)P L _!h)i=n,B=O,b=p;}N=L;}
n+=J||(g=I+p,m=p<O?g-3:g+2,*m<z|m[O-p]||I[p+=p-O]);}}}}Z!r&q>2||(p=O,q|A>2|o>z&
!r&&++C*--A));}}}Z++O>98?O=20:e-O);P N+M*M&&N>-K+1924|d?N:0;}main(){Z++B<121)*G
++=B/x%x<2|B%x<2?7:B/x&4?0:*l++&31;Z B=19){Z B++<99)putchar(B%x?l[B[I]|16]:x)_
x-(B=F)){i=I[B+=(x-F)*x]&z;b=F;b+=(x-F)*x;Z x-(*G=F))i=*G^8^y;}else v u,5);v u,
1);}}
```

Summary

- Functions as data is **one of the two (programming) big ideas** in this course
- It's a beautiful example of the **abstraction of the list iteration details**

(Image Credit: *Simply Scheme* by Brian Harvey & Matt Wright)

Turning function machines into plowshares