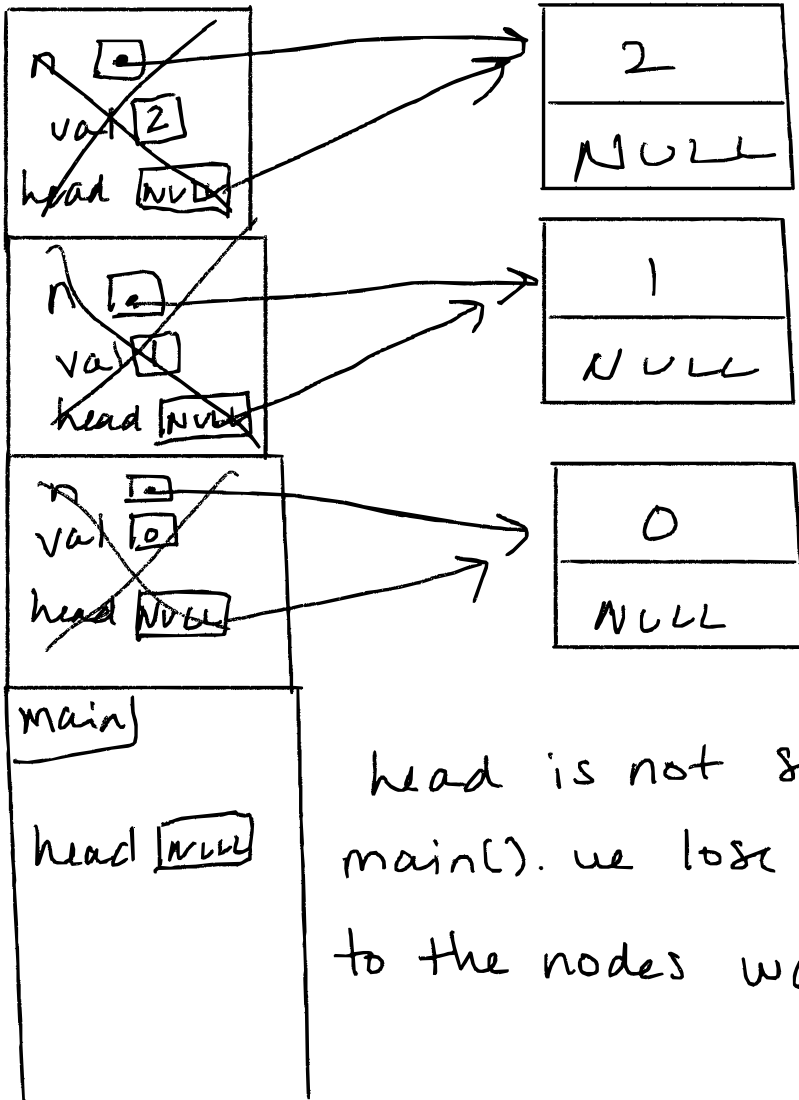


# Dynamic LL V2

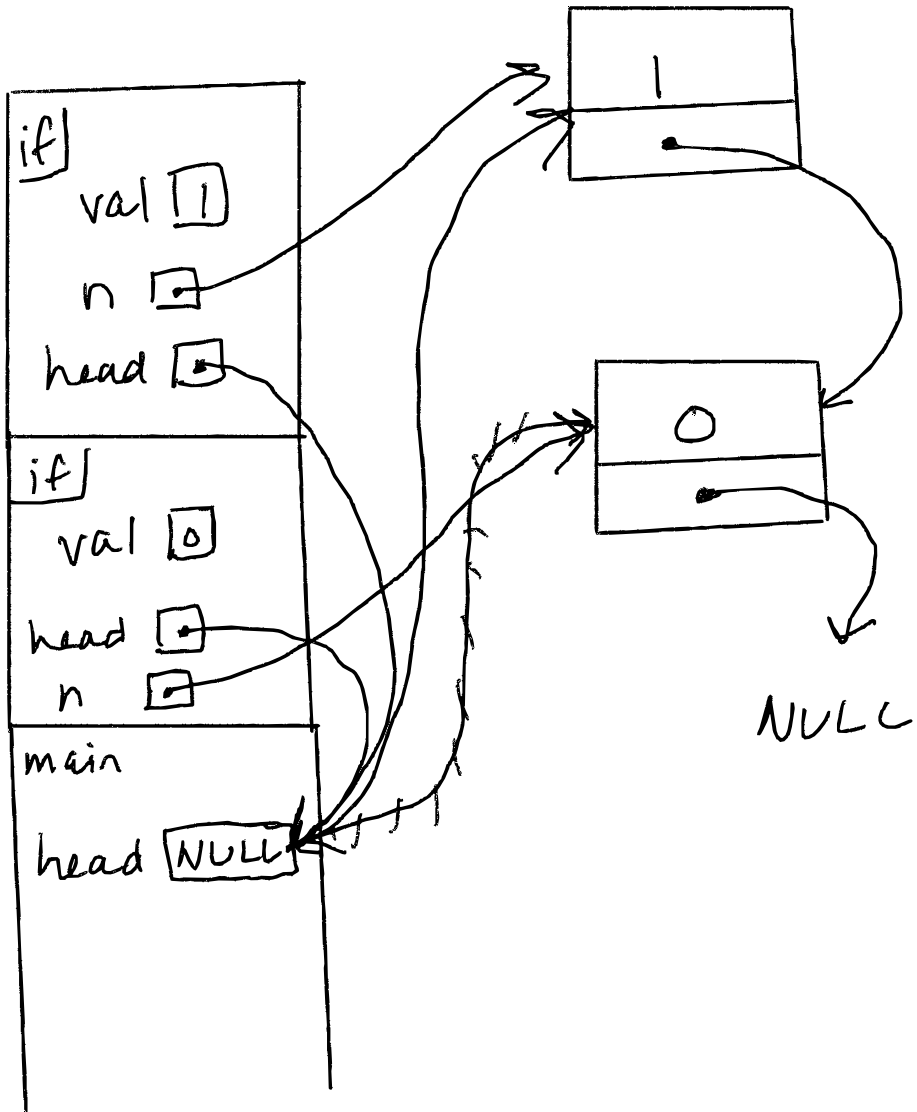
This doesn't work. why?



head is not set in  
main(). we lose access  
to the nodes we create.

# Dynamic LL v2 (fixed)

$n \rightarrow \text{next} = \text{head} \rightarrow \text{ptr?}$



# Binary and Data Representation

\* 8 bits = 1 byte

\* bytes are the smallest units of data that we can read/write

\* word : fixed (default) unit of memory that hardware works with

↳ depends on processor/arch

↳ typically 32 or 64 bytes

why use variables?

why use types?

variables are basically labels  
that reference memory

types indicate:

- 1) how much memory is associated with the variable (1 byte? 4 bytes?)
- 2) how to interpret the data (ex: 4 bytes. is it an int? float?)

## Ex: Simple Image

|          |           |
|----------|-----------|
| 00 white | 01 orange |
| 10 blue  | 11 black  |

2 bits to represent

4 colors

byte representation is 12 bytes

# Number Bases & Unsigned Integers

Decimal numbers:

$$5163 = 5 \times 10^3 + 1 \times 10^2 + 6 \times 10^1 + 3 \times 10^0$$

$$= \overset{d_3}{5}000 + \overset{d_2}{1}00 + \overset{d_1}{6}0 + \overset{d_0}{3}$$

General Formula

$$\sum_{i=0} d_i \times 10^i$$

# Notation

$0b$  (zero b)  $\rightarrow$  binary number

$0x$  (zero x)  $\rightarrow$  hexadecimal number  
(base 16)

$0$  (zero)  $\rightarrow$  octal number  
(base 8)

$d_0$   $\rightarrow$  lowest order bit

$d_{N-1}$   $\rightarrow$  highest order bit

# Hexadecimal (Base 16)

compact way to represent  
binary numbers

↳ each single hex digit  
represents 4 binary bits,  
allowing large binary  
strings to be written in  
a fraction of the space

| Dec | Bin  | Hex |
|-----|------|-----|
| 0   | 0000 | 0x0 |
| 1   | 0001 | 0x1 |
| 2   | 0010 | 0x2 |
| 3   | 0011 | 0x3 |
| 4   | 0100 | 0x4 |
| 5   | 0101 | 0x5 |
| 6   | 0110 | 0x6 |
| 7   | 0111 | 0x7 |
| 8   | 1000 | 0x8 |
| 9   | 1001 | 0x9 |
| 10  | 1010 | 0xA |
| 11  | 1011 | 0xB |
| 12  | 1100 | 0xC |
| 13  | 1101 | 0xD |
| 14  | 1110 | 0xE |
| 15  | 1111 | 0xF |

0b10

$$2 = 1 \times 2^1 + 0 \times 2^0$$

$$2^3 = 8$$

0b100

# Exercise : Hexadecimal

convert binary to hex:

01010011000110111101001100  
4 C 6 F 4 C  
(12) (15)

$$2^2 + 2^3 \\ = 4 + 8 = 12$$

$$2^0 + 2^1 + 2^2 + 2^3 \\ = 1 + 2 + 4 + 8 \\ = 15$$

0x4C6F4C

# Exercise : Hexadecimal

Convert Hex to Binary:

0x 8045EF

0b 1000 0000 0100 1110 1111

E  
14

F  
15

$$2^3 + 2^2 + 2^1$$
$$8 + 4 + 2 = 14$$

$$2^3 + 2^2 + 2^1 + 2^0$$
$$= 8 + 4 + 2 + 1$$
$$= 15$$

# Exercise: Hexadecimal to Decimal

0x3CD0


$$3 \times 16^3 + C \times 16^2 + D \times 16^1 + 0 \times 16^0$$

$$= 3 \times 16^3 + 12 \times 16^2 + 13 \times 16^1 + 0 \times 16^0$$

$$= 15568$$

# Decimal to Hexadecimal

for  $i$  from  $N-1$  to  $0$  do:

$$i^{\text{th}} \text{ digit} = \text{floor}\left(\frac{\text{input}}{16^i}\right)$$

$$\text{input} = \text{input} \% 16^i$$

$$\begin{array}{r} 9742 \\ \hline 16 \end{array} \quad R: 14 \quad E$$

9742<sub>10</sub>  
to hex

$$\begin{array}{r} 608 \\ \hline 16 \end{array} \quad R: 0$$

$$\begin{array}{r} 38 \\ \hline 16 \end{array} \quad R: 6$$

$$\begin{array}{r} 2 \\ \hline 16 \end{array} \quad R: 2$$

$$\begin{array}{r} 0 \\ \hline 16 \end{array} \quad R: 0$$



0x260E

# Decimal to Binary

can use the same approach,  
but there is an easier way:

Repeatedly divide by 2

and check the parity

(even or odd)

for  $i$  from 0 to  $N-1$

$$i^{\text{th}} \text{ digit} = \text{input} \% 2$$

$$\text{input} = \text{floor} \left( \frac{\text{input}}{2} \right)$$

Exercise: Convert  $422_{10}$  to binary ↓ decimal notation

$$\frac{422}{2} \rightarrow 0 \quad (\text{lowest order})$$

$$\frac{211}{2} \rightarrow 1 \quad 205.5$$

$$\frac{105}{2} \rightarrow 1 \quad 52.5$$

$$\frac{52}{2} \rightarrow 0$$

$$\frac{26}{2} \rightarrow 0 \quad 13.5$$

$$\frac{13}{2} \rightarrow 1 \quad 6.5$$

$$\frac{6}{2} \rightarrow 0$$

$$\frac{3}{2} \rightarrow 1 \quad 1.5$$

$$\frac{1}{2} \rightarrow 1$$

$$0 \rightarrow 0$$

highest order

0.110100110

# Coding with Powers of 2

ex:  $1 \ll 3$  means:

take binary representation of 1  
and shift all the bits

3 places to the left

$$1 = 0b\underbrace{0001}_{\text{uu}} \rightarrow 0b1000 \quad (8)$$

ex:  $2 \ll 1$

$$2 = 0b0\underbrace{10}_{\text{u}} \rightarrow 0b0100 \quad (4)$$

Shortcut :  $1 \ll n = 2^n$



$$2 < < 3$$

010010

1 0 0 0 0

$$1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$$

$$5 < 2$$

$$0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$

$$0101$$

$$010100$$

$$2^4 + 2^2 = 16 + 4 = 20$$

# Purpose of Bit Shifting

Shortcut for multiplying or  
dividing numbers by 2, 4, 8, 16...  
done at the lowest, fastest  
level a computer understands

bit masks, permissions, flags,  
packed data, binary formats,  
low-level hardware...

# Base 10 Analogy

45

- shift digits left one space  
(relative to decimal point)

45.0 →

450.

- shift digits right one space  
(relative to decimal point)

4.5 0

4.5

so 4

# Sizes of Data Types

What factors should we consider?

- 1) amount of space needed to store
- 2) range of values it supports

# Unsigned Integers

What is the largest number that can be stored in 4 bits?

$$\begin{aligned} 0b1111 &= 2^3 + 2^2 + 2^1 + 2^0 \\ &= 8 + 4 + 2 + 1 = \underline{\underline{15}} \end{aligned}$$

What is the largest number that can be stored in 4 bytes?

$$4 \text{ bytes} = 32 \text{ bits}$$

$$\hookrightarrow \text{so } 2^{32} - 1 \quad \left( \begin{array}{c} \text{in general:} \\ 2^N - 1 \end{array} \right)$$

Why minus one?

if you have  $2^{32}$  total values, and you start counting at 0, the highest value is  $2^{32} - 1$

# Overflow

Ex: When we try to store the value 16 in 4 bit number, we get the value 0.

10000

Ex: Storing 20 in a 4 bit value gives

$2^4$   ~~$2^3$~~   $2^2$   ~~$2^1$~~   $2^0$

1 0 1 0 0  
          

4

What is the output of  
this program?

infinite loop!

It will go 5, 4, 3, 2, 1, 0...

And then wrap around to  
the largest possible unsigned  
integer!

because unsigned ints can  
never be negative

# Two's Complement

Example: What is 1001  
as a signed integer?

$$\begin{aligned} & -2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 2^0 \\ & = -8 + 1 = -7 \end{aligned}$$

as an unsigned integer?

$$\begin{aligned} & 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 2^0 \\ & = 8 + 1 = 9 \end{aligned}$$

Ex: Convert 0b0110  
(signed) to decimal

$$\begin{aligned} & -0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 \\ & = 4 + 2 = 6 \end{aligned}$$

Ex: convert 0b1111 (signed)  
to decimal

$$\begin{aligned} & -1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 \\ & = -8 + 4 + 2 + 1 \\ & = -8 + 7 = -1 \end{aligned}$$

Two's Complement : 4 bits

$$\text{Ex: } 7 + 3 = -6 \text{ (overflow)}$$

$$\text{Ex: } -8 - 1 = 7 \text{ (overflow)}$$

$N$  bits: range of unsigned integers we can represent?

$$[0, 2^N - 1]$$

$N$  bits: range of signed integers we can represent?

$$[-2^{N-1}, 2^{N-1} - 1]$$

Example: 2 bits

---

$$[-2^{n-1}, 2^{n-1} - 1]$$

$$n=2$$

$$[-2, 1]$$

00

0

01

1

10

-2

11

-1

# Negate a two's complement signed integer

- flip bits
- add one

Ex: -5 from 5 in binary

0101

↓

1010

↓

1011

$$-2^3 + 2^1 + 2^0 = -8 + 3 = -5$$

# Binary Addition

2 + 8 as unsigned  
binary numbers

$$\begin{array}{r} 0010 \\ + 1000 \\ \hline 1010 \end{array}$$

$$2 + 8 = 10$$

↪  $2^3 + 2^1 = 8 + 2 = 10$

# Binary Addition : Overflow

What happens when we run out of bits? Truncation!

12 + 7 (as unsigned 4 bit nums)

12 + 7 should be 19

$$\begin{array}{r} \overset{1}{0}b\ \overset{1}{1}\ 00 \\ +\ 0b\ 0\ 1\ 1 \\ \hline 1b\ 0\ 0\ 11 \\ \uparrow \text{overflow} \end{array}$$

