CSS 342 - Mathematical Principles of Computing clearing transaction
- Data structures, maintainable, good quality code. data → memory
- dude uses a scriptor to grade → strict on naming, style conventions idempotent
- 5 projects → start early! C++ has pitfalls Shields uses G++ compiler
- char strings end in "∅" ⇒ test code outside of Visual Studio constants
zero out memory contents or face unpleasant consequences in header files
globals vs locals { use getters
Interpreter vs. Compiler Structs require semi-colons
CSC 332 ints have different bit sizes in C++
Use message board for 332 or even 342 Questions
email Nash if needed

CSS 342 C: Mathematical Principles of Computing p.4/4

Canvas/Catalyst EPost (Message Board):
You can access the message board link on the class website to exchange messages with your classmates. Please use this board for only discussions. No junk email. Note that the professor will not keep track of all messages. The professor monitors this board once a day Mon.& Wed.

Topics covered and tentative schedule: This is an approximate ordering of topics. Material will take about the allotted time; we do not cover all sections in all chapters. See Website for updates. Readings prefixed with C++ or Math correspond to the appropriate textbook.

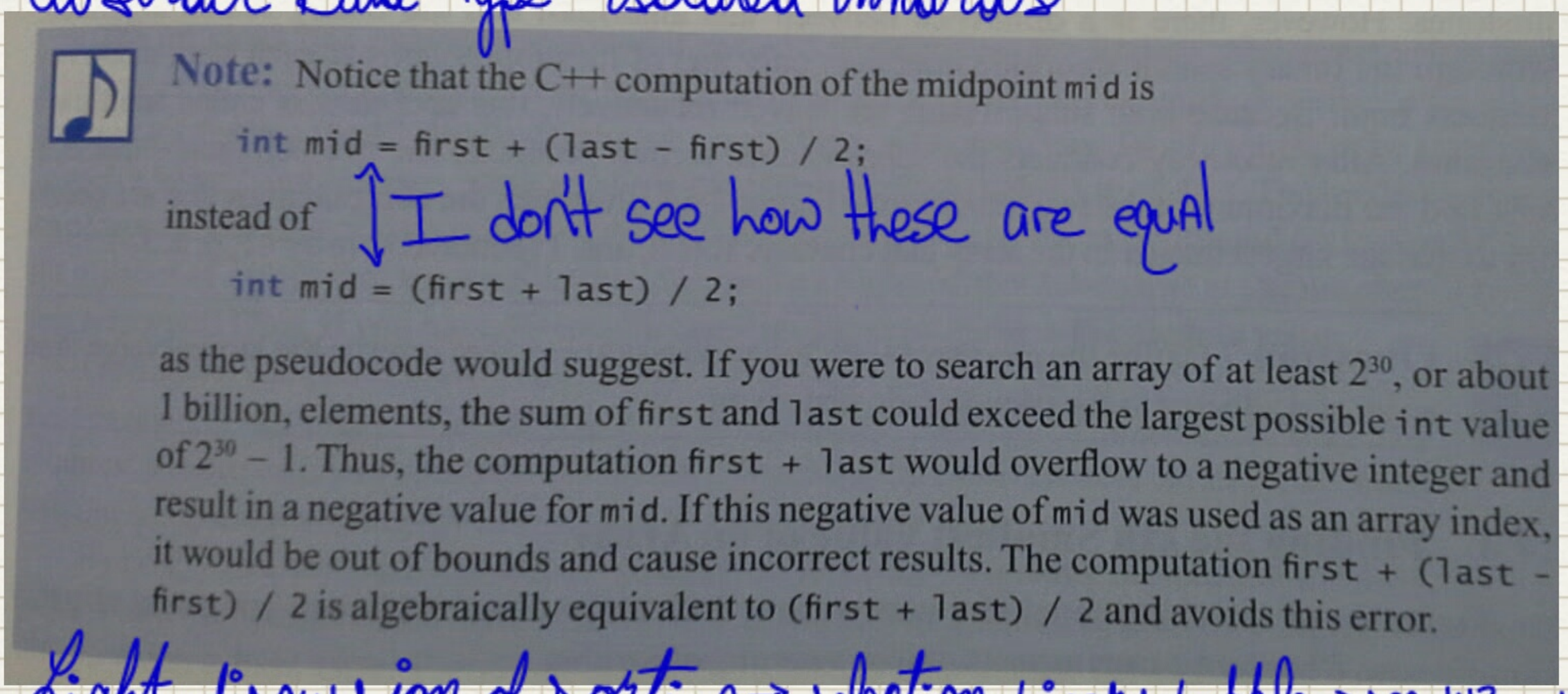
week	Date	**Topic** (topics and work are subject to change by instructor as needed)	**Read this Material before class** 6th ed. chapters for each book are in ()	**Lab(Program) Due**
0	Sep 25	Introduction, etc. C++ language structure, C++ structs, arrays	Website C++ Notes, C++ appendix A, G,H,I,K,J as needed	
1	Sep 30	Software Engineering Principles, pair programming; Searching, simple note: p.71 (L-f)/2sorting (selection, bubble, insertion)	C++ 1, 9.2 (6ed. appendices BCDF, 2.1, 2.4.2, 10.2.5, 11.1, interlude 3 for asset, appx I for doxygen, G for file I/O, L for library includes), Website Notes	
	Oct 2	Data Abstraction (Objects and Classes), C++ classes	Website C++ Notes C++ 3 (6ed. 1, Interlude 1))	
2	7	More C++ classes	"	lab 1 due
	9	Pointers, Linked Lists	C++ 4.1-4.4 (6ed. Interlude2; ch.4,8,914.1.2)	
3	14	more Pointers, Linked	Website	lab 2 due
	16	Templates, STL intro	C++ 4.5, apx E (6ed. 1,8.3, Interlude 1 & 7)	
4	21	Algorithm analysis	C++ 9.1 (6ed. 10, 11.3) ?Math 2.4, 3.1-3.3	
	23	Algorithm analysis continued		
5	28	Recursion	C++ 2 (6ed. 2, 3.3, 4.3, 5, 6.5, 9.2.3) ?Math 5.3-5.4 (4.3-4.4 ed. 6)	lab 3 due
	30	**Midterm exam**		
6	Nov 4	Sorting (using recursion)	C++ 9.2 (some of) (6ed. 11)	
	6	Binary Search Tree intro	C++ 10.3 (6ed. 15, 16), Website links/notes ?Math 11.1-11.3 (10.1-10.3 ed. 6)	
7	11	No school - holiday		
	12	**Last day to drop a course (& a late fee is assessed)**		
	13	Object-oriented design, Stacks	C++ 1, 7 (6ed. 1, 6, 7)	lab 4 due
8	18	Queues	C++ 7 (6ed. 13,14)	
	20	Recurrence and Induction	C++ 2.1, 5.3 (6ed. 2.2,2.3, 5.4, appendix E), ?Math 5.1-5.2 (4.1-4.2 ed. 6)	lab 5 design due
9	25	Recurrence and Induction continued	C++ appendix D ?Math 8.1, 8.3 (7.1, 7.3 ed. 6)	
	27	Representation of Integers	?Math 4.2 (3.6 ed. 6) Website links/notes	
10	Dec 2	Mathematical Foundation	?Math 1.1, 1.3-1.5 (1.1-1.4 ed. 6)	
	4	Intro to Design Patterns, last day stuff	Website Course Notes	lab 5 due
11	9	**Final exam**, Monday, in class		

Appendix G - File I/O → Read doxygen - page 777
H - Some standard headers
K - From Java to C++

Object-Oriented Solution
use classes → cohesive modules and loosely coupled modules
e.g. read module, sort module, print module

Operation contract → docs use specify pre- and post- conditions

review slides
Abstract Data Type - isolated innards



Note: Notice that the C++ computation of the midpoint `mid` is

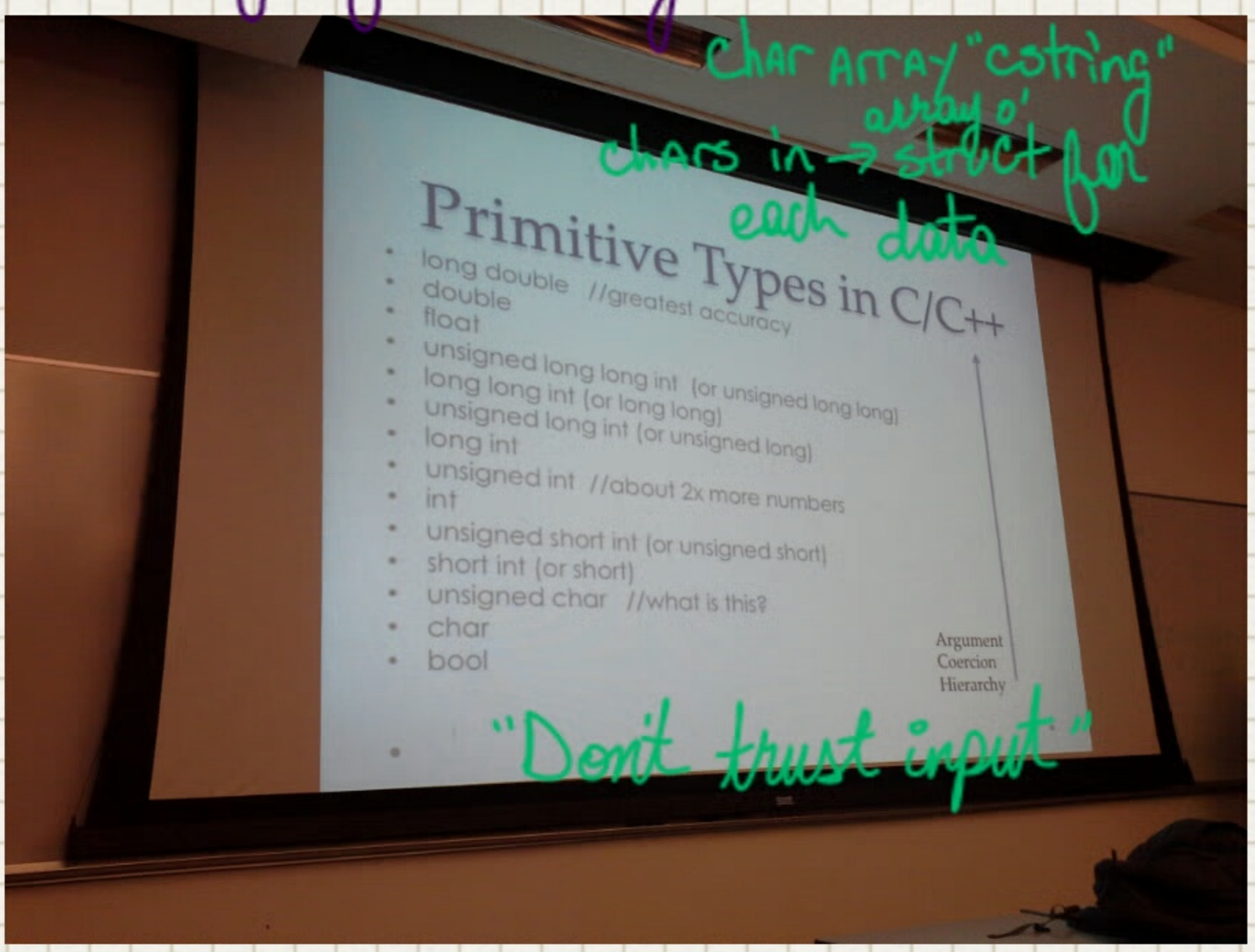
```
int mid = first + (last - first) / 2;
```

instead of ↕ I don't see how these are equal

```
int mid = (first + last) / 2;
```

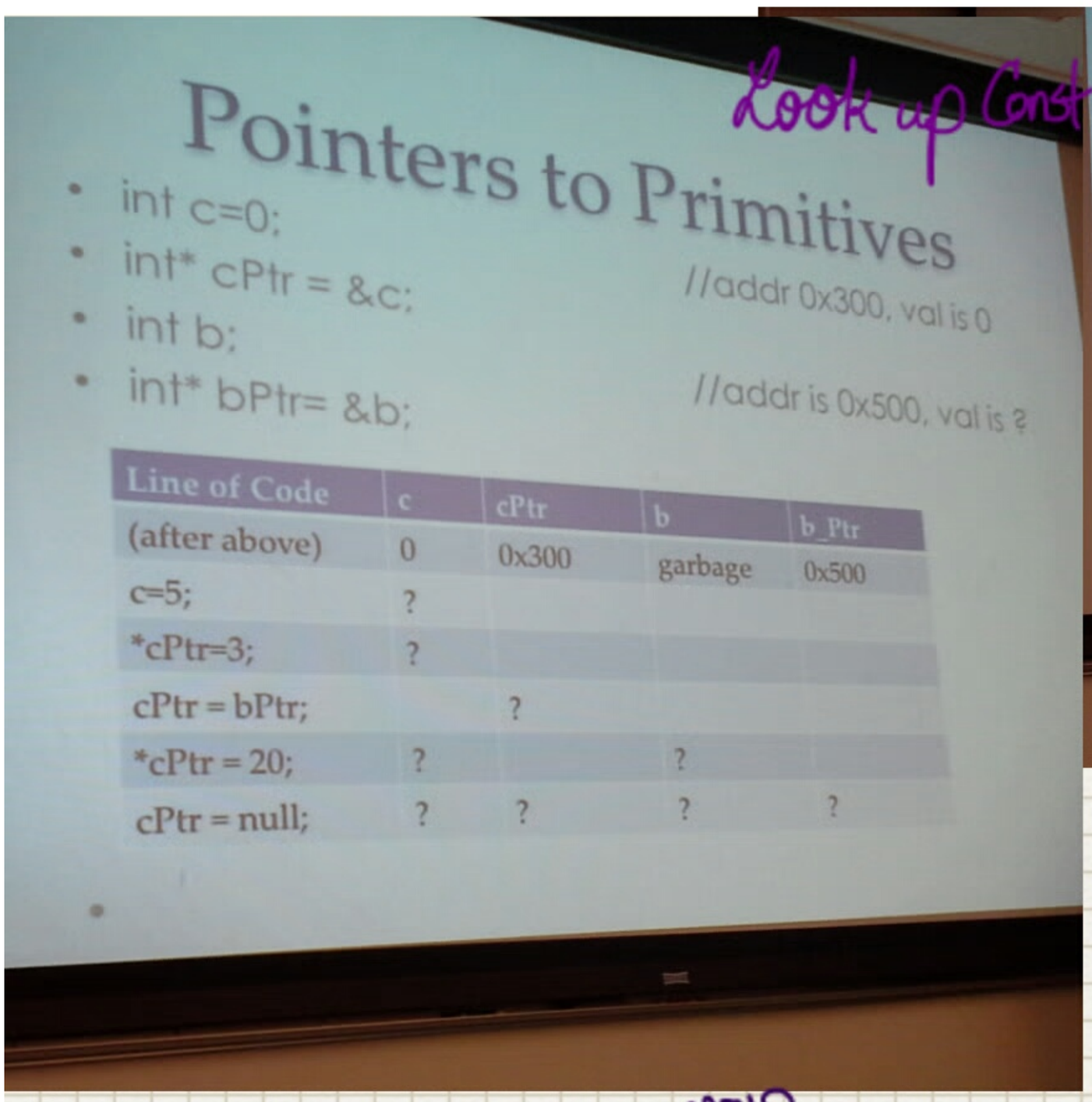
as the pseudocode would suggest. If you were to search an array of at least 2^{30}, or about 1 billion, elements, the sum of `first` and `last` could exceed the largest possible `int` value of $2^{30} - 1$. Thus, the computation `first + last` would overflow to a negative integer and result in a negative value for `mid`. If this negative value of `mid` was used as an array index, it would be out of bounds and cause incorrect results. The computation `first + (last - first) / 2` is algebraically equivalent to `(first + last) / 2` and avoids this error.

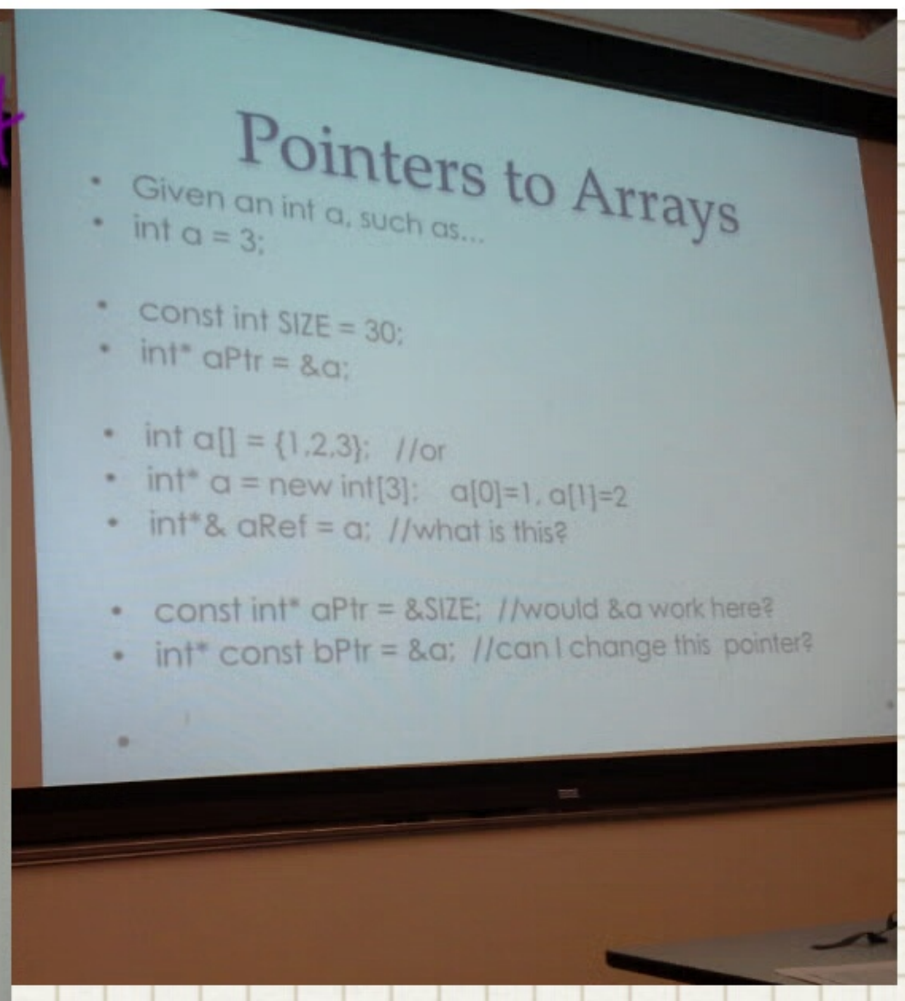
Light discussion of sorting → selection, binary, bubble → see 143 notes
structs for first assignment. Utility functions (Decompose project #1
headers declare mains dependency divide & conquer



char array "cstring" array o' chars in → struct for each data

When passing in char array use an array of structs
Use string copy





97 53 10
x 97 53 10

9 7 5 3 10
79 35 01
01 35 79

11,10,9,8,7,6,5,4,3,2,1,0 M=12
[11,10,9,8,7,6] [5,4,3,2,1,0] ①
[11,10,9] [8,7,6] [5,4,3] [2,1,0] ②
[11,10] [9] [8,7] [6] [5,4] [3] [2,1] [0] ③
[11][10] [8][7] [5][4] [2][1] ④

$[11,10,9,8,7,6,5,4,3,2,1,0] = A_0$

$[0,0,0,0,0,0,0,0,0,0,0,0] = B_0$ 0=make B

$[11,10,9,8,7,6,0,0,0,0,0,0] = B_1$

$[0,0,0,0,0,0,5,4,3,2,1,\emptyset\ ] = A_1$ 1=A→B

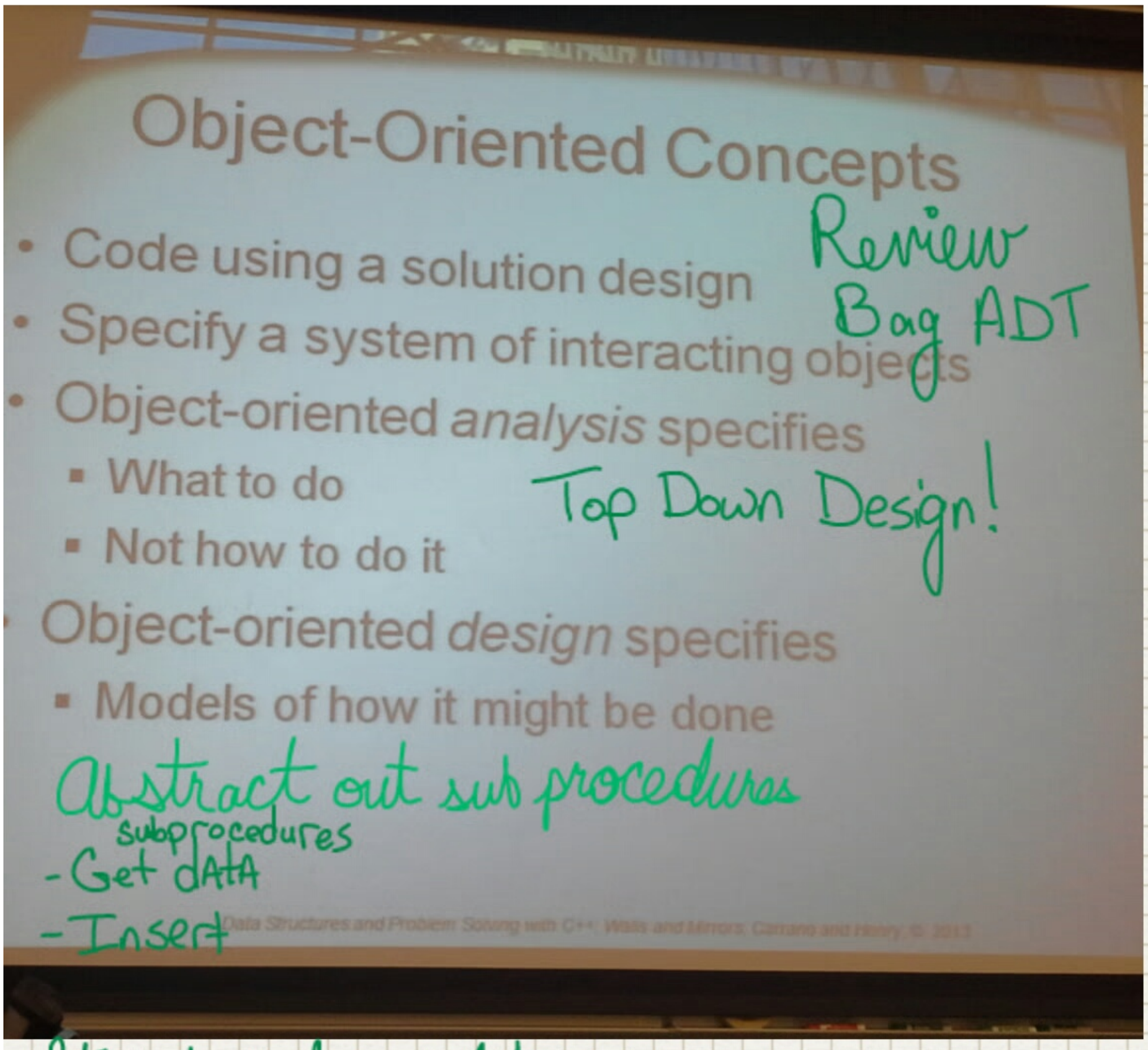
$[0,0,0,8,7,6,0,0,0,0,0,0] = B_2$
$[11,10,9,0,0,0,5,4,3,2,1,\emptyset\ ] = A_2$ } 2=B→A, 4=B→A

$[11,10,0,8,7,6,0,0,0,0,0,0] = B_3$ 3=A→B

$[0,0,9,0,0,0,5,4,3,2,1,\emptyset\ ] A_3$

$[11,0,9,0,0,0,5,4,3,2,1,\emptyset] = A_4$

[0 10 0



Steps to develop a solution:

1. Object Oriented Analysis → "Discovery" of problem. Not focusing on soln
   - What does the end user expect?
   - Requirements ⇒ what a soln must be and do.
   - Express the problem and requirements of the soln in terms of objects
   - Describe these objects and their interactions

Default Constructor broke equality → didn't initialize

Interlude 1: Classes

- If a method does not change an object, mark it as const
  - source in a copy method.

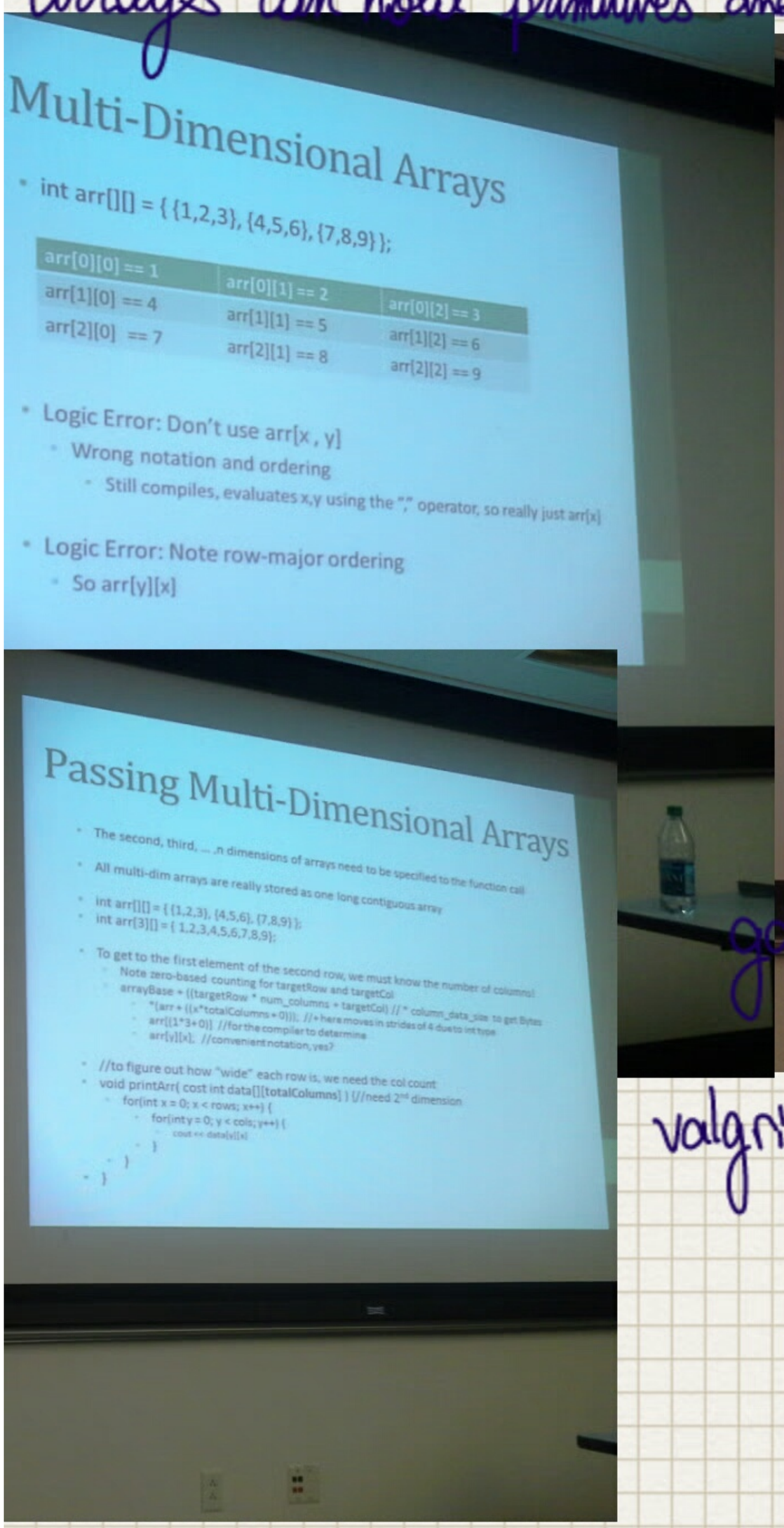
p.40 → Inheritance

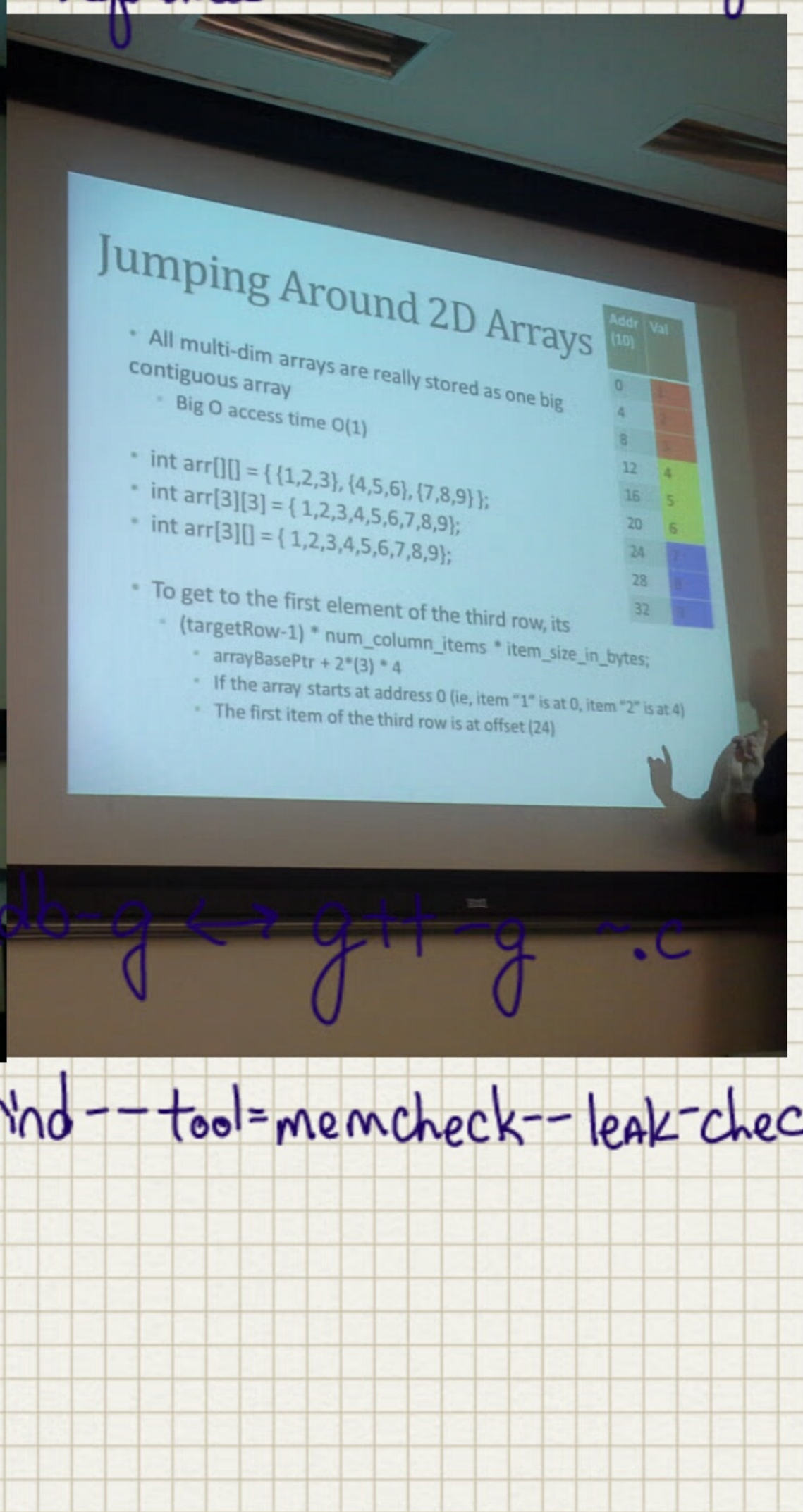
plainbox examples

Linked lists in C++ always lives on heap

Dynamic data structures live on heap

Arrays can hold primitives and references → Make our own better versions
→ Bounds checking, deep copies
Automatic size Mgmt, dynamic





gdb -g ↔ g++ -g ~.c

valgrind --tool=memcheck --leak-check=yes

$3y' + ty = 4 \quad P(t) = \frac{t}{3}$

$y' + \frac{t}{3}y = \frac{4}{3} \quad \mu(t) = e^{\int \frac{t}{3} dt}$

$\mu(t) = e^{t^2/6}$

$e^{t^2/6} y' + \frac{t e^{t^2/6}}{3} = \frac{4}{3} e^{t^2/6} \leftrightarrow \int \frac{d}{dt}\left[e^{t^2/6} y\right] dt = \int \frac{4}{3} e^{t^2/6} dt \leftrightarrow y \approx \frac{4}{3} e^{-t^2/6} \int_0^t e^{s^2/6} ds$

$\frac{4}{3}\int e^{\frac{t^2}{6}} dt$

LoL DFQ

Look @ Nash's Stack main.cpp Chapter 20 in Soustrup

kill destructor, copy, assign

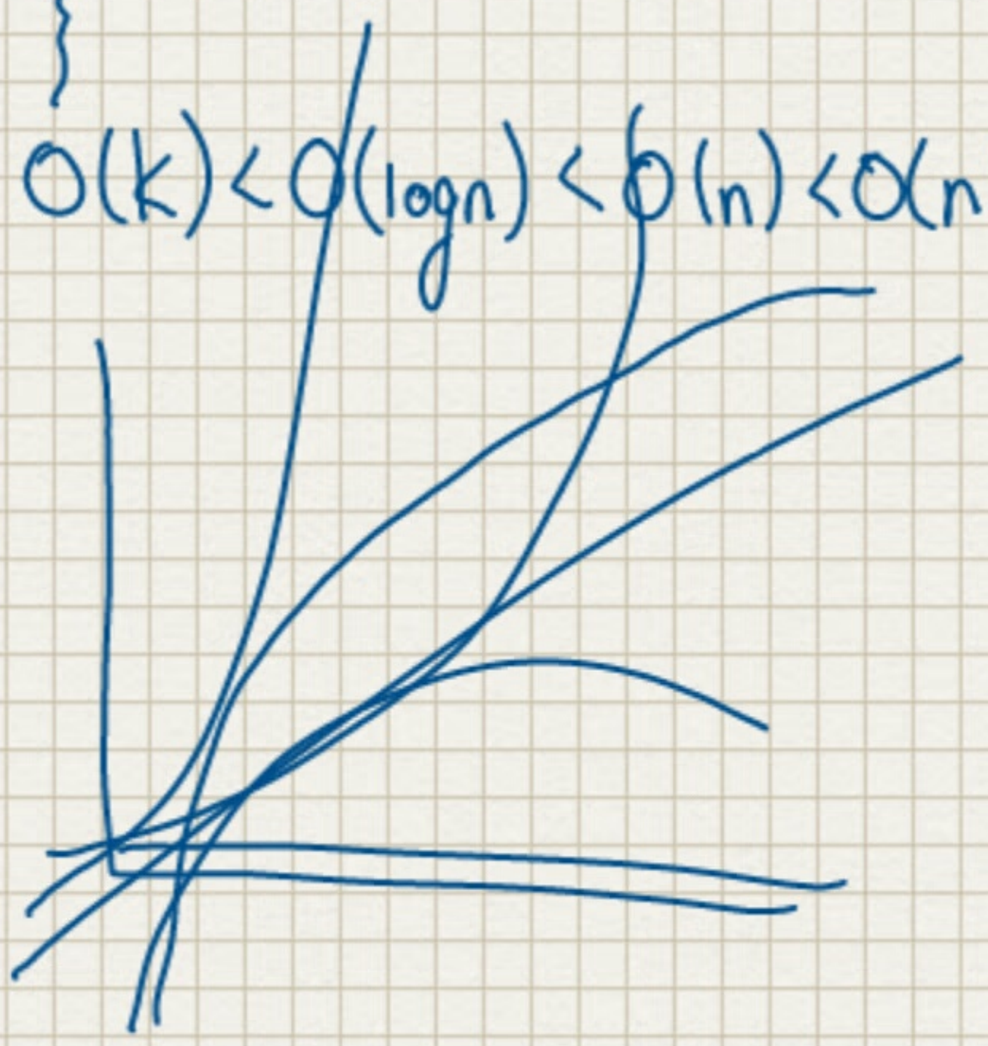
Could you help me write the employee destructor for me?

Employee: ~Employee(){

}

While n!=0

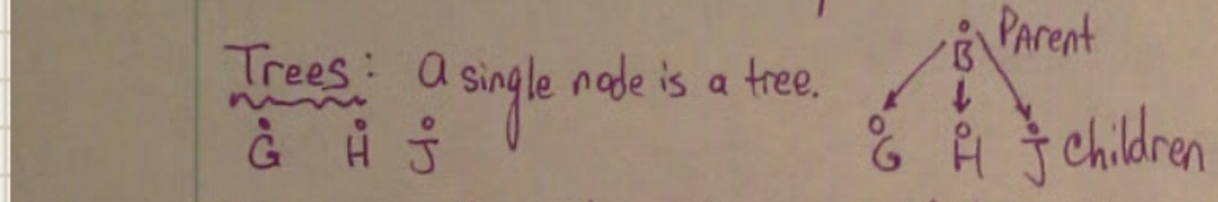
$O(k) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3) < O(2^n) < O(N!)$



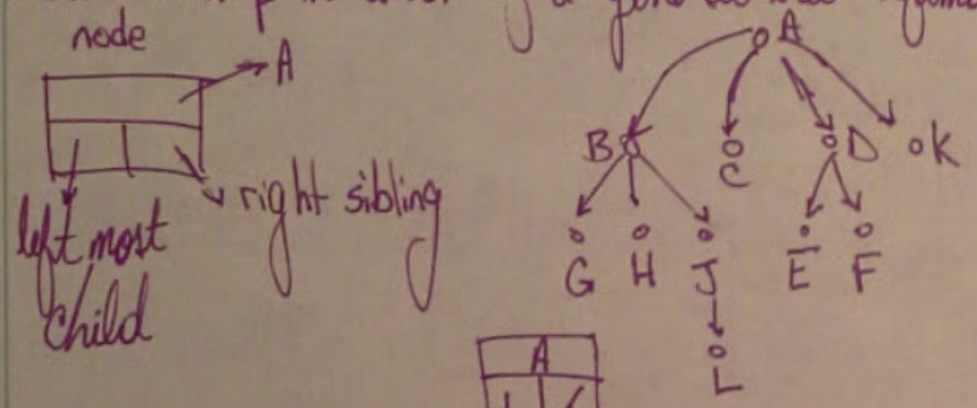
Disclaim unreliable methods.

○ Bubblesort for comment how-to. What code is doing → not how.
end column comments → keep it short

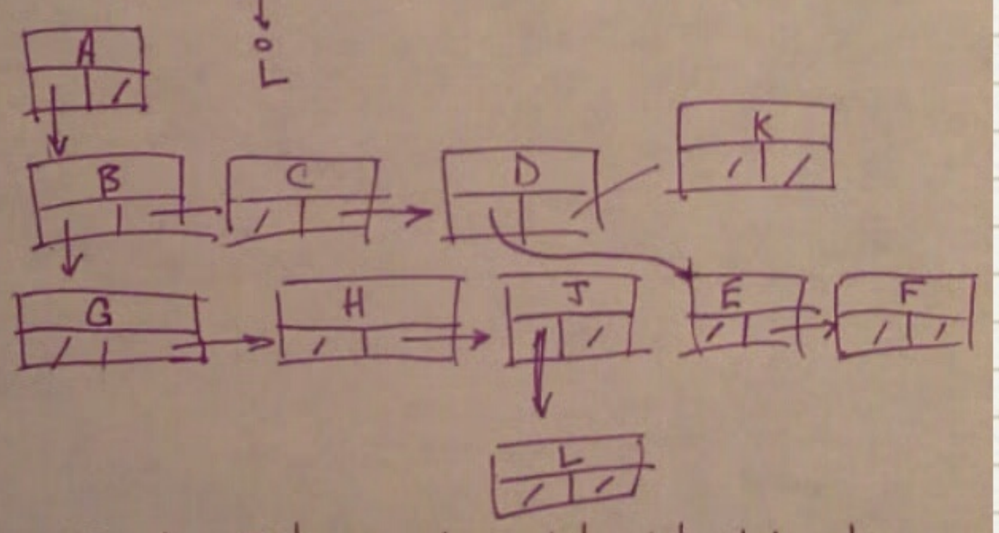
Trees: a single node is a tree.



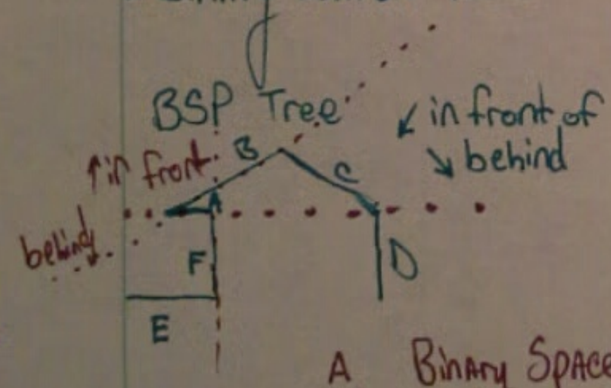
Common implementation of a general tree - leftmost child, right sibling

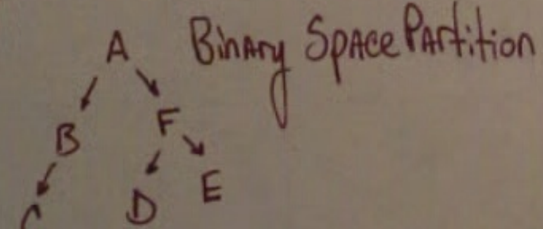


- Expression Tree
- BSP Tree
- Huffman encoding algorithm
- Binary Search Tree

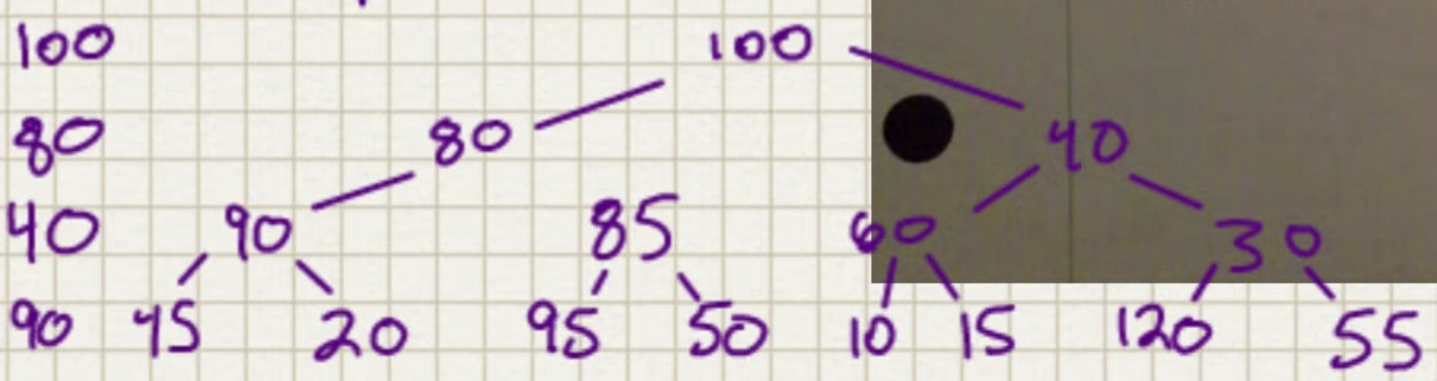


Encoding - Find a uniquely encoded and decoded code s.t. chars w/ higher freq have shorter encodings



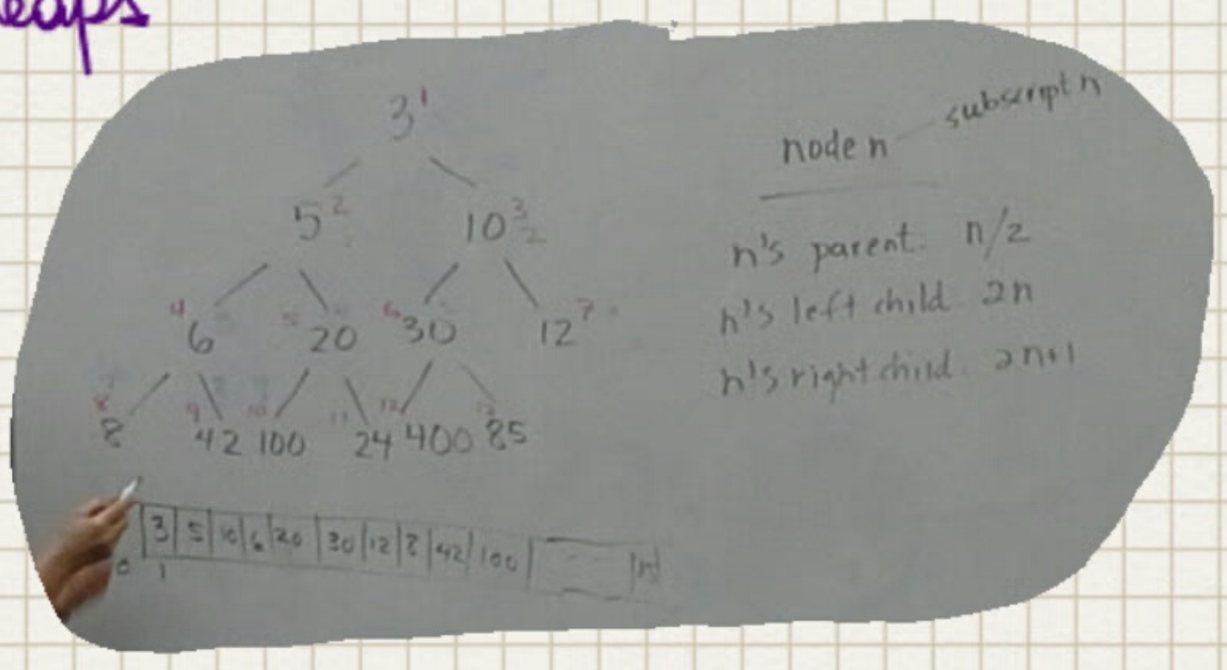


Build heap in linear time
must have data up-front
data - no particular order



sort/heapify lowest level
- compare children, swap lowest
- little trees are heaps
- go up a level

sort next level up

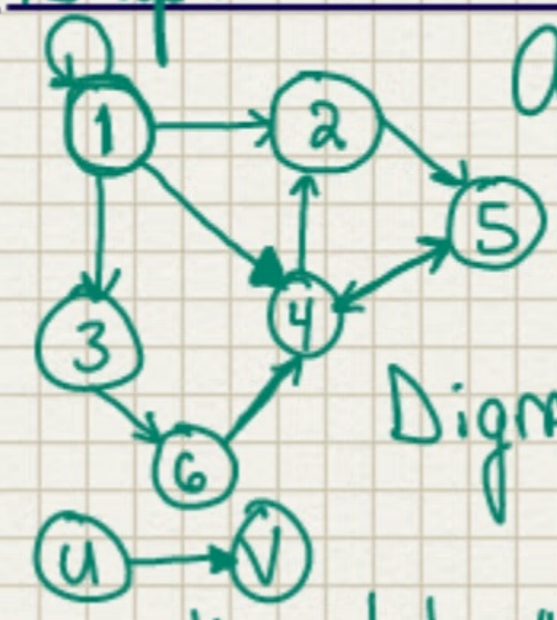


January 22, 2014 CS343 Lab 1 trends: ./a.out > file2
- comment all data members & functions
- over use of set/get
- output
- molest loop increment steps + mess w/ bounds.
- Test getter/setters
- Linux → diff file1 file2

Graphs

a Graph: (N, E) → set of edges
↓ set of nodes

- edges connect vertices
- edge is an ordered pair (u,v)



Digraphs

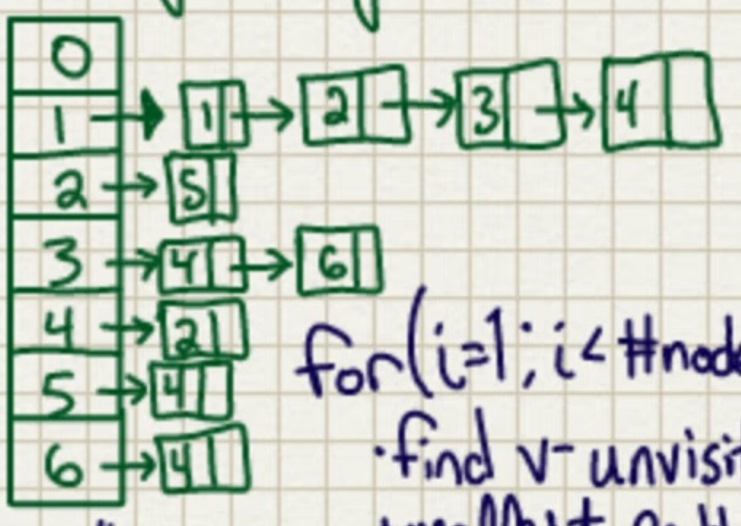
"v is adjacent to u"

Implementation:

1. adjacency matrix = A → # of paths of length 1
   A² → # of paths of length 2
   A³ → # of " " " 3

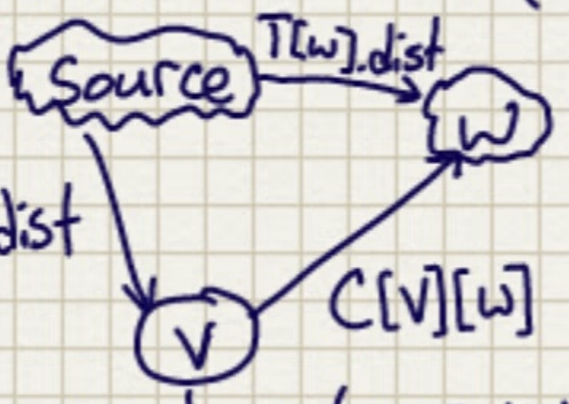
	1	2	3	4	5	6
1	1	1	1	1	0	0
2	0	0	0	0	1	0
3	0	0	0	1	0	1
4	0	1	0	0	1	0
5	0	0	0	1	0	0
6	0	0	0	1	0	0

2. Adjacency lists



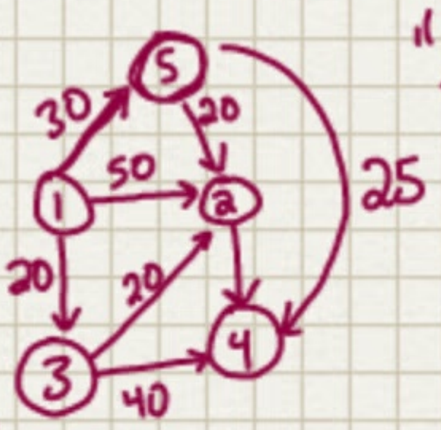
for(i=1; i<#nodes-1; i++){
- find v - unvisitted with smallest path distance.
- mark v visitted
- for each w adjacent to v
  if(w is not visitted){

T[w] = min(T[w].dist, T[v].dist + C[v][w])



Dijkstra's shortest path

- finds the shortest distance from a source node to every other node.



"Greedy Algorithm"

- Consider the source
- Consider lowest cost edges
- mark lowest edge vertex
- Try next-lowest-edge vertex
- Try to use other paths to try to get there faster

every node gets a chance to be the source

- Given a source.

C	1	2	3	4	5
1	-	50	20	-	30
2	-	-	-	10	-
3	-	20	-	40	-
4	-	-	-	-	-
5	-	20	-	25	-

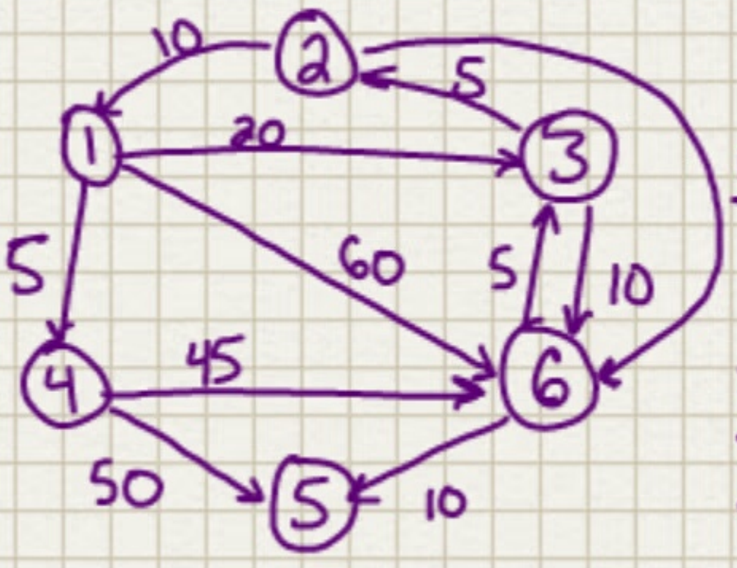
Cost Matrix

T[w/e].dist

1	2	3	4	5
0	∞	∞	∞	∞

1/27/2014

hold a bool to validate your graph - bool says if graph has changed.
↳ ADT → a graph class → Dijkstra's just a method.



source = 1

	T[w].dist	1	2	3	4	5	6
		0	∞	∞	∞	∞	∞
V=1		∞	20	5	∞	∞	60
V=4		∞	20		55		50
V=3		25			55		30
V=2					55		30
V=6					40		30

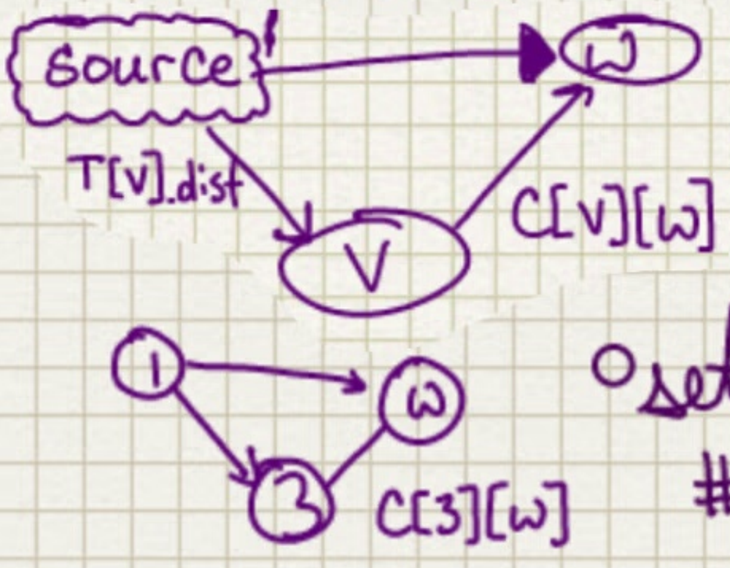
0	25	20	5	40	30
1	2	3	4	5	6

T[w].PATH

	1	2	3	4	5	6
	0	0	0	0	0	0
V=1	0	0	1	1	0	1
V=4	0	0	1		4	4
	3				4	3
					6	5
						3

0	3	1	1	6	3
1	2	3	4	5	6

T[w].dist
↳ cost from source to w



Want path from 1 to 5

5
T[5].path=6
T[6].path=3
T[3].path=1 → We can stop myeAhh.
T[1] path=0 → 1,3,6,5

- set width

```
#include<iomanip>
cout<<setw(5)<<10
-----10
----100
```

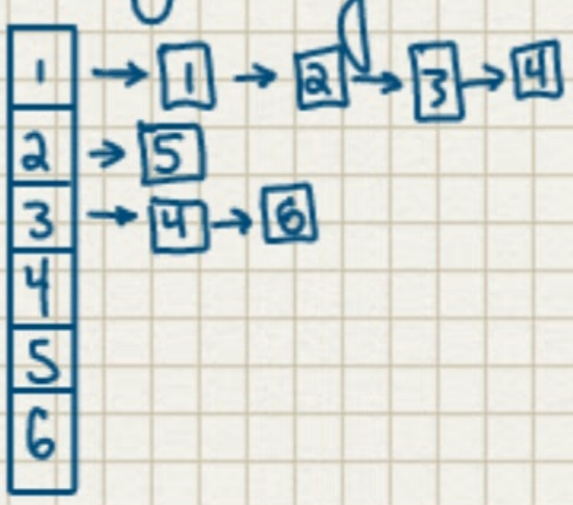
from to to

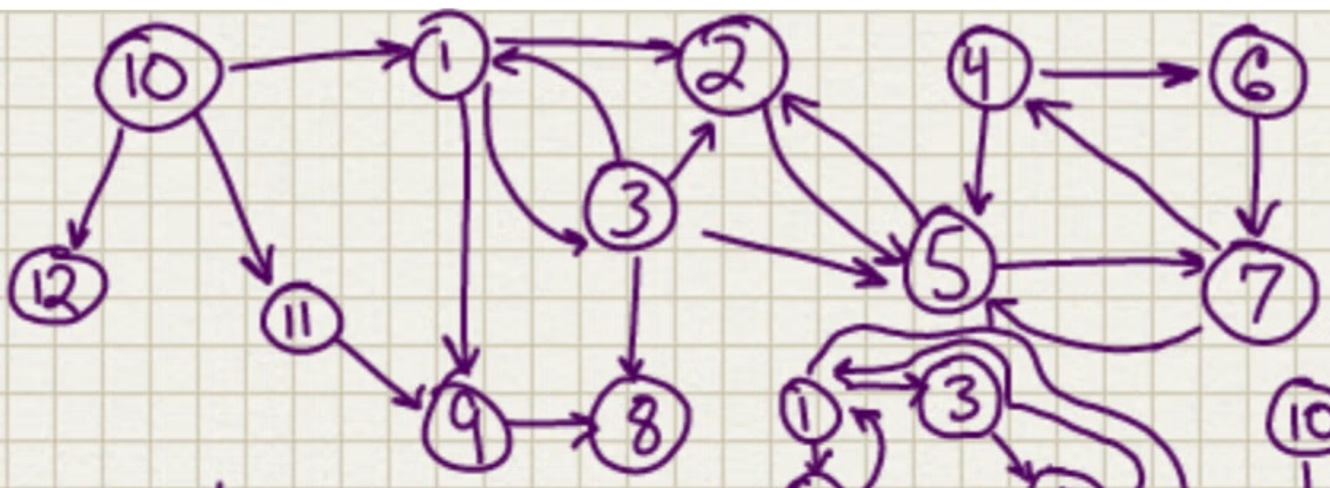
T[TO].PATH=C
T[C].PATH=B
T[B].Path=A
T[A].Path=From

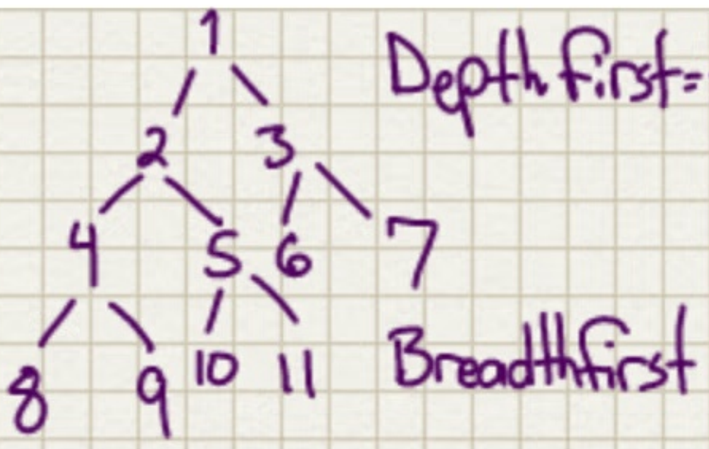
From A B C TO

```
int i=to,
cur=T[From][TO].Path;
cout<<curr
cur=T[From][cur].Path
```

2. Adjacency List

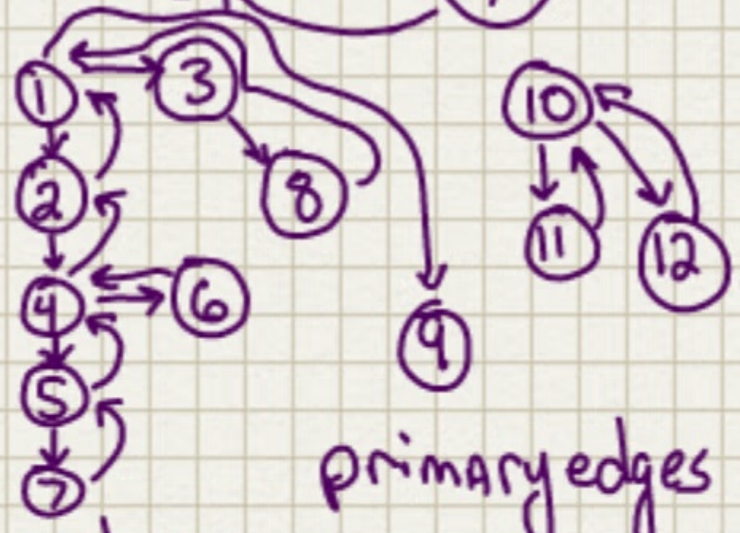






Depth first = Preorder

Breadth first



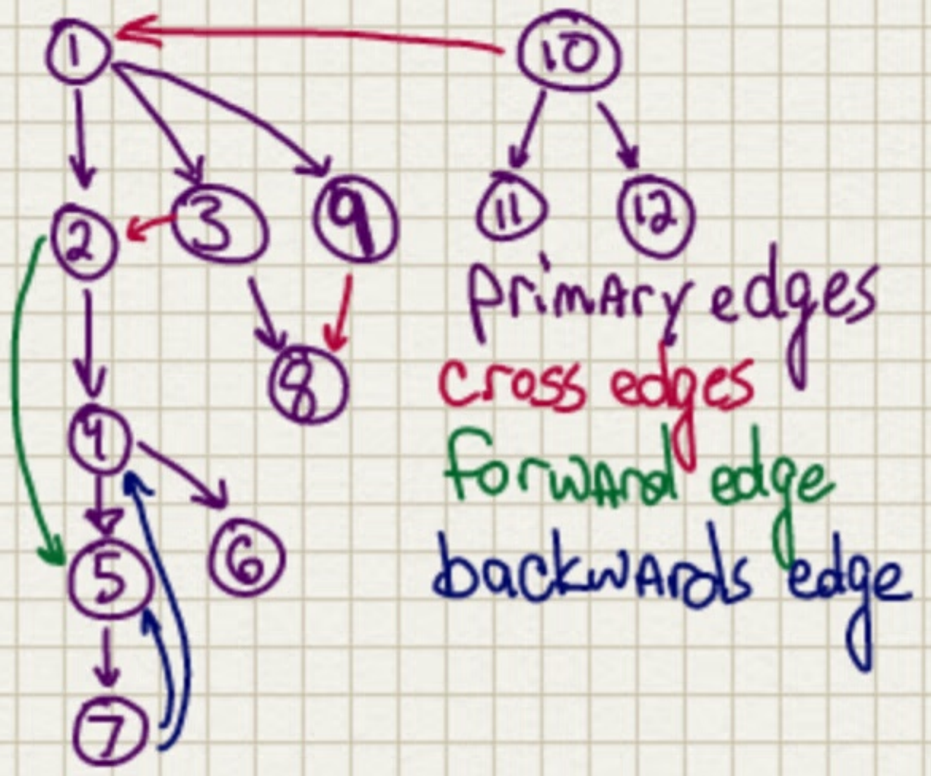
done

primary edges down

```
void depthfirstSearch{
    for (v=1; v<n; v++){
      if(v not visitted)
          dfs(v);
void dfs( v     )
    mark v visitted.
    do whatever with v
    for each w adjacent to V
        if(w not visitted)
            dfs(w)
```

DFS order

1 2 4 5 7 6 3 8 9 10 11 12



primary edges

cross edges

forward edge

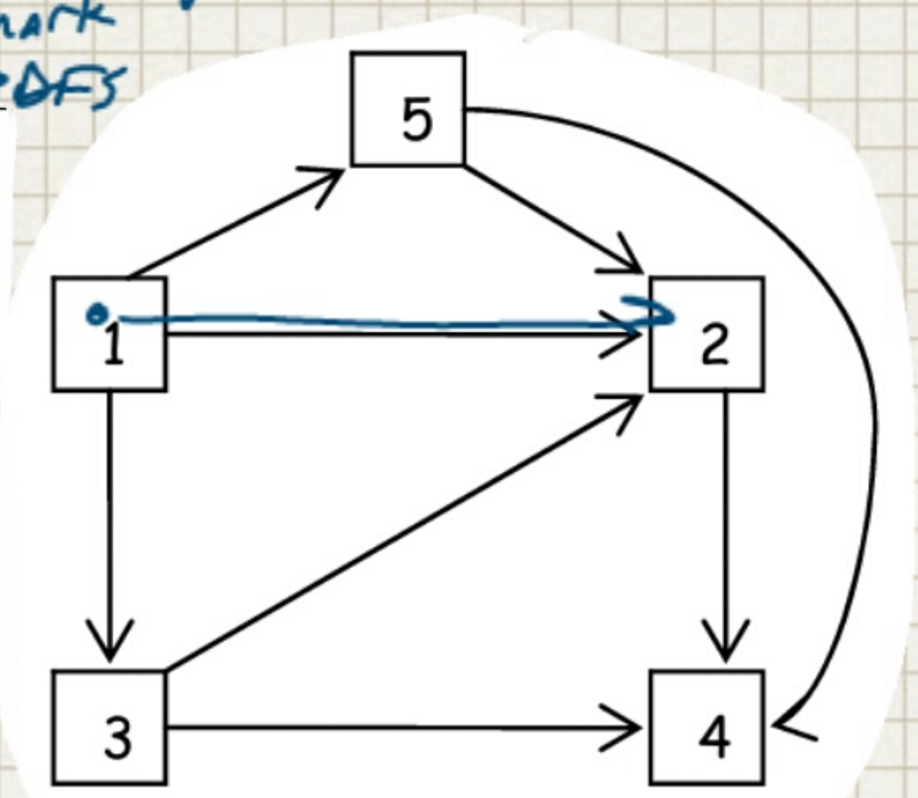
backwards edge

Breadth First Search

```
void bfs(v )
    mark v visitted
  q.enQueue(v)
    while q not empty
        x= q.dequeue();
        do whatever with x.
        for each w adjacent to x.
            if(w not visitted)
                mark w visitted
                q.enque(w)
```

start at first node defined → 1
→ last edge of this node
→ mark
→ DFS

```
5
Aurora and 85th
Green Lake Starbucks
Woodland Park Zoo
Troll under bridge
PCC
1 5
1 3
1 2
2 4
3 4
3 2
5 4
5 2
0 0
```



12435

unsigned problem →
constructor →
zero poly → not printing zero

Test Cases
- negative exponents
- negative coeffs

Read!

Don't overcomplicate → follow Zander's more closely

1/29/2014

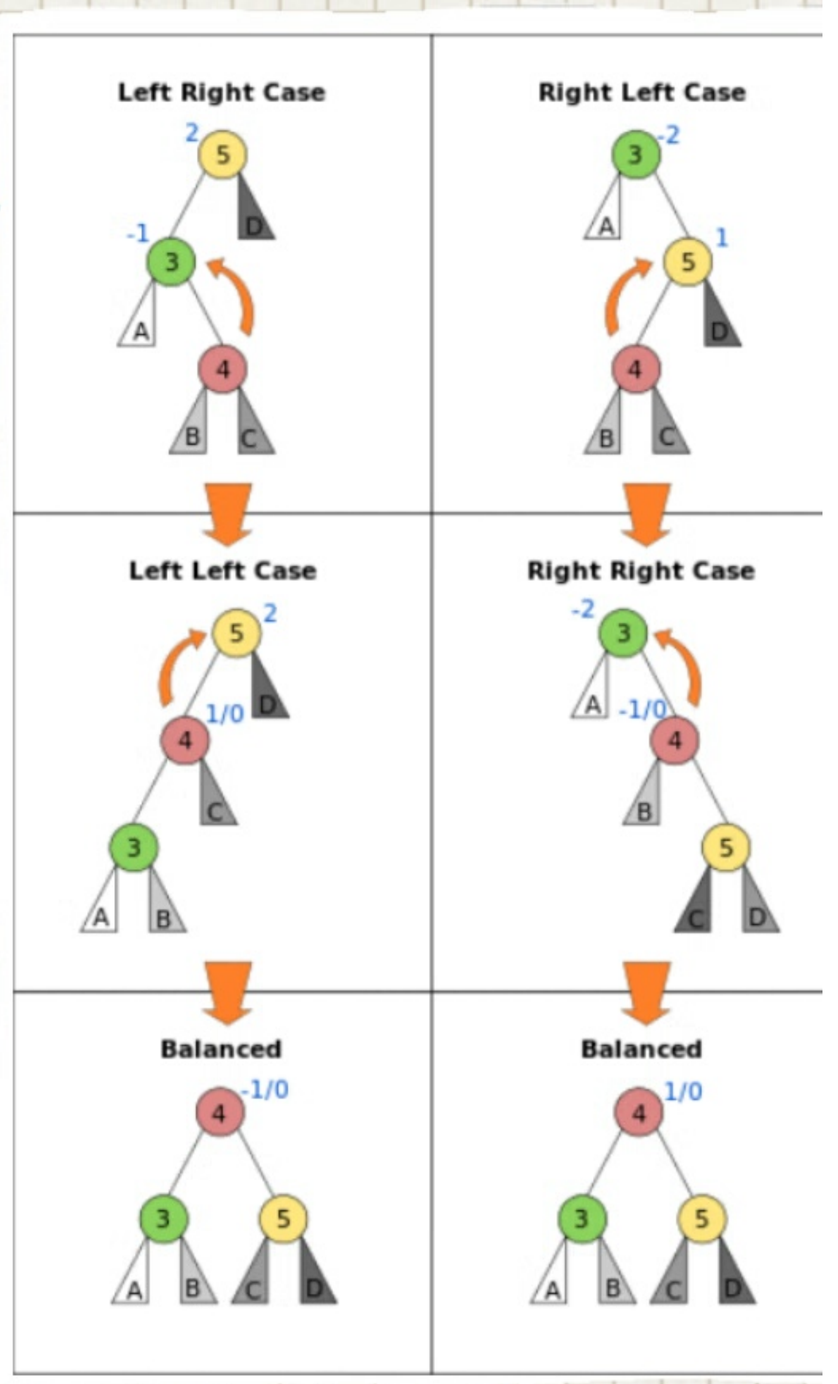
AVL Trees → Definition of Balanced: ∀ node ∈ tree, the height of it's left and right subtree differs by at most one.

→ For each insert, you do at most one single or double rotation

"Rotations": 4 kinds:

→ single rotation
- left left
- right right

→ double rotations
- left right
- right left



http://en.m.wikipedia.org/wiki/AVL_tree

Huffman coding

Trees - BSP
- Huffman Coding
- Binary Search tree

10 points? 10 minutes

20 points? 20 minutes!
↳ Actually "take" practice exam.

Poly class → know the code
↳ review Labs 1 & 2

Execution tree!!!!!
↳ recursion on paper

Heaps → Problem based ↳ short answer
↳ not too much code ↳ Data structures

Graphs → Dijkstra's → "Dijkstra doesn't work on negative edges"

Depth first → AVL insert "by hand"

Breadth first

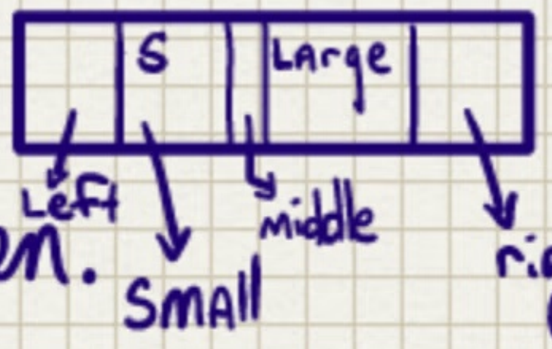
Balance Trees ZOMG Complexities!
↳ AVL + 2/3 + "Tris"
↳ not too codey → insert → left-left
↳ Up to 1/3/2014

2-3 tree
↳ B-tree: Balanced trees
- le root is either a leaf or has between 2 and m children.

All non-leaf nodes have between $\lceil \frac{m}{2} \rceil$ and m children.

2-3 trees

insert is Typically recursive.



All nodes are leaves or have 2 or 3 children.

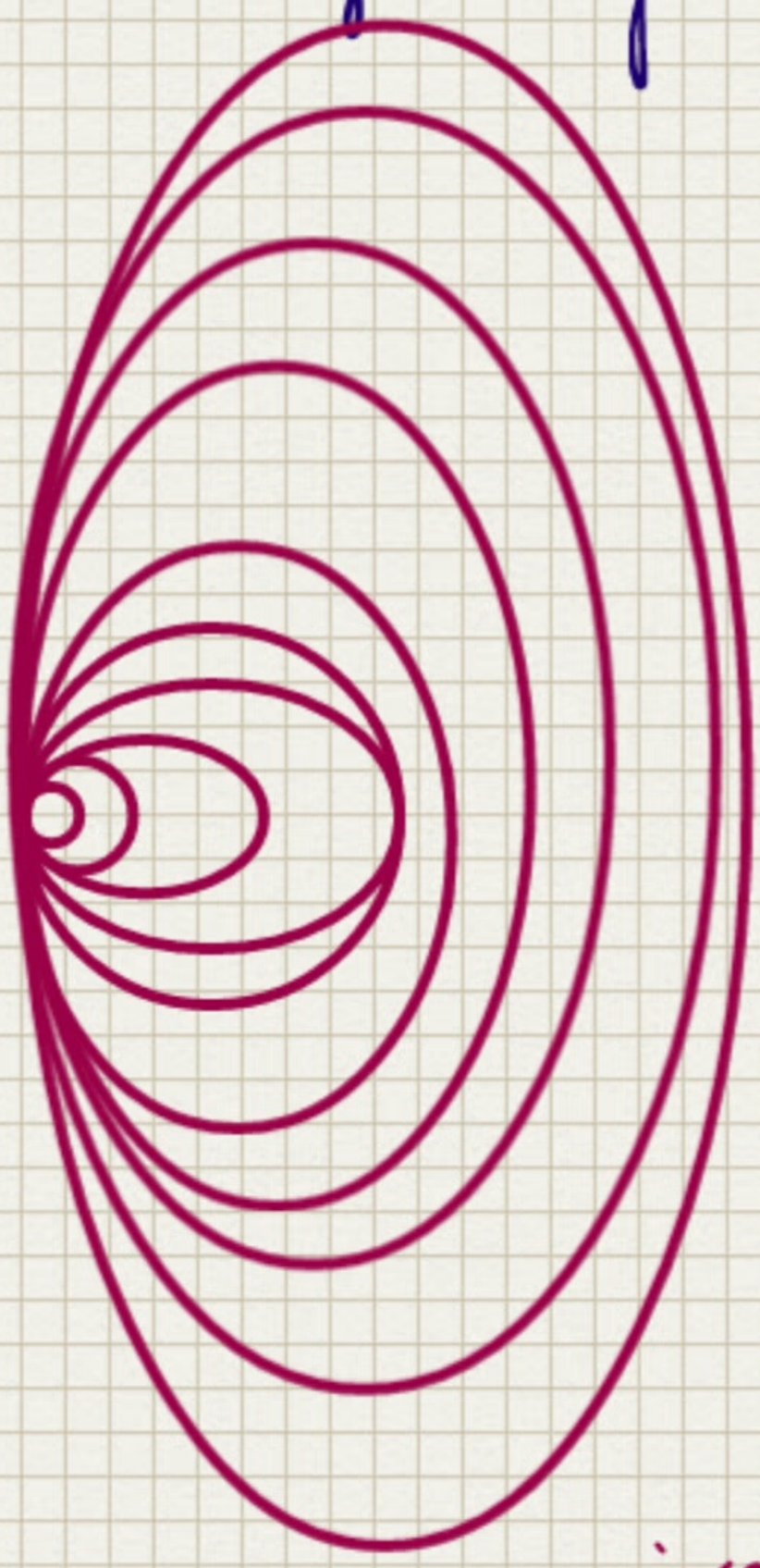
→ node has only 1 value → 2 children → right pointer = nuLL.

→ node has 2 values → has 3 children.

Items that are smaller go in left subtree

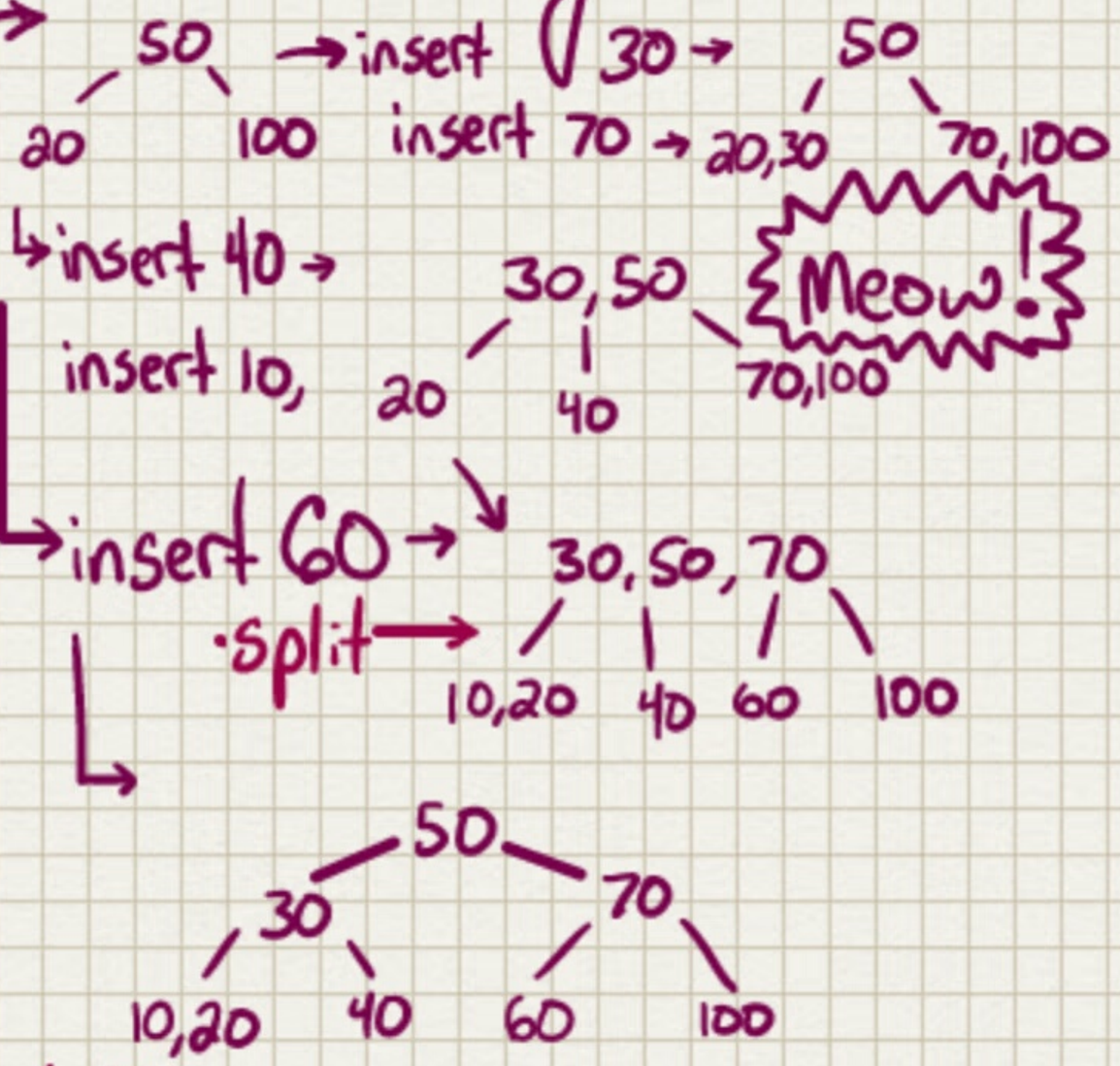
Items >S, <L go in middle subtree

Items >L go in right subtree.



- insert → 50 → 50
- ↳ 100 → 50,100
- ↳ 20 → "split"

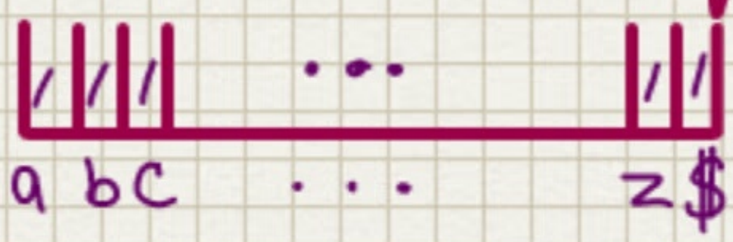
1. middle value "moves up"
2. surrounding nodes become children

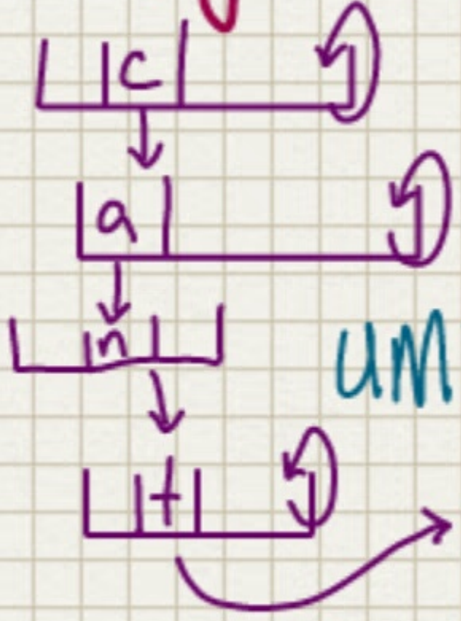


Trie (reTRIEval)

one way to implement a dictionary (or any n-bit string)

empty trie:





UML Learns

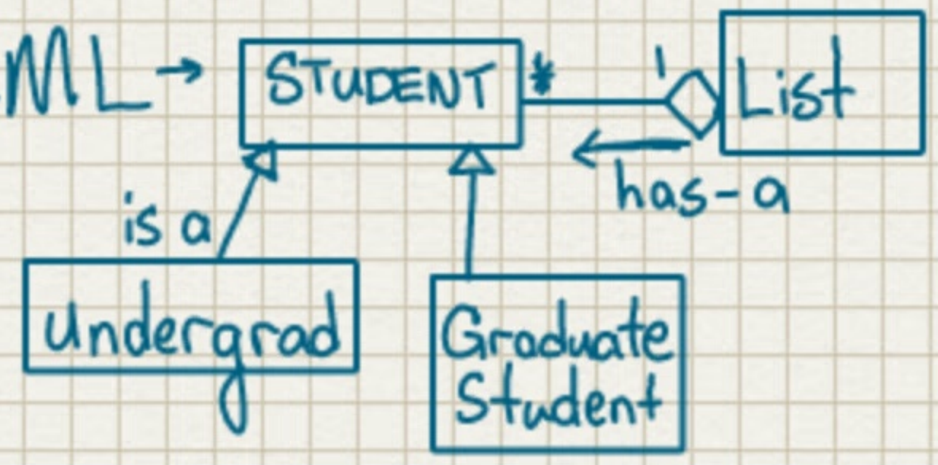
has-a composition

Insert for a trie → Hashing → if NULL new node* → update cur to pt to next.

ptr to array

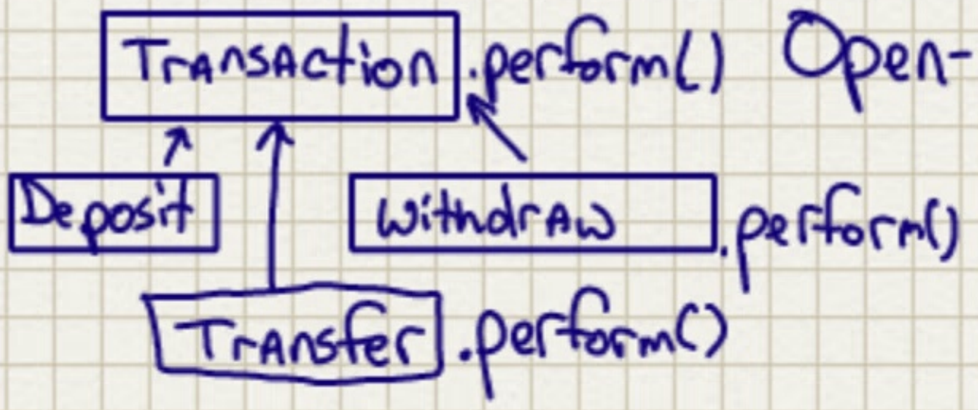
Inheritance → NOT Different in C++

UML →



Aggregation/Composition

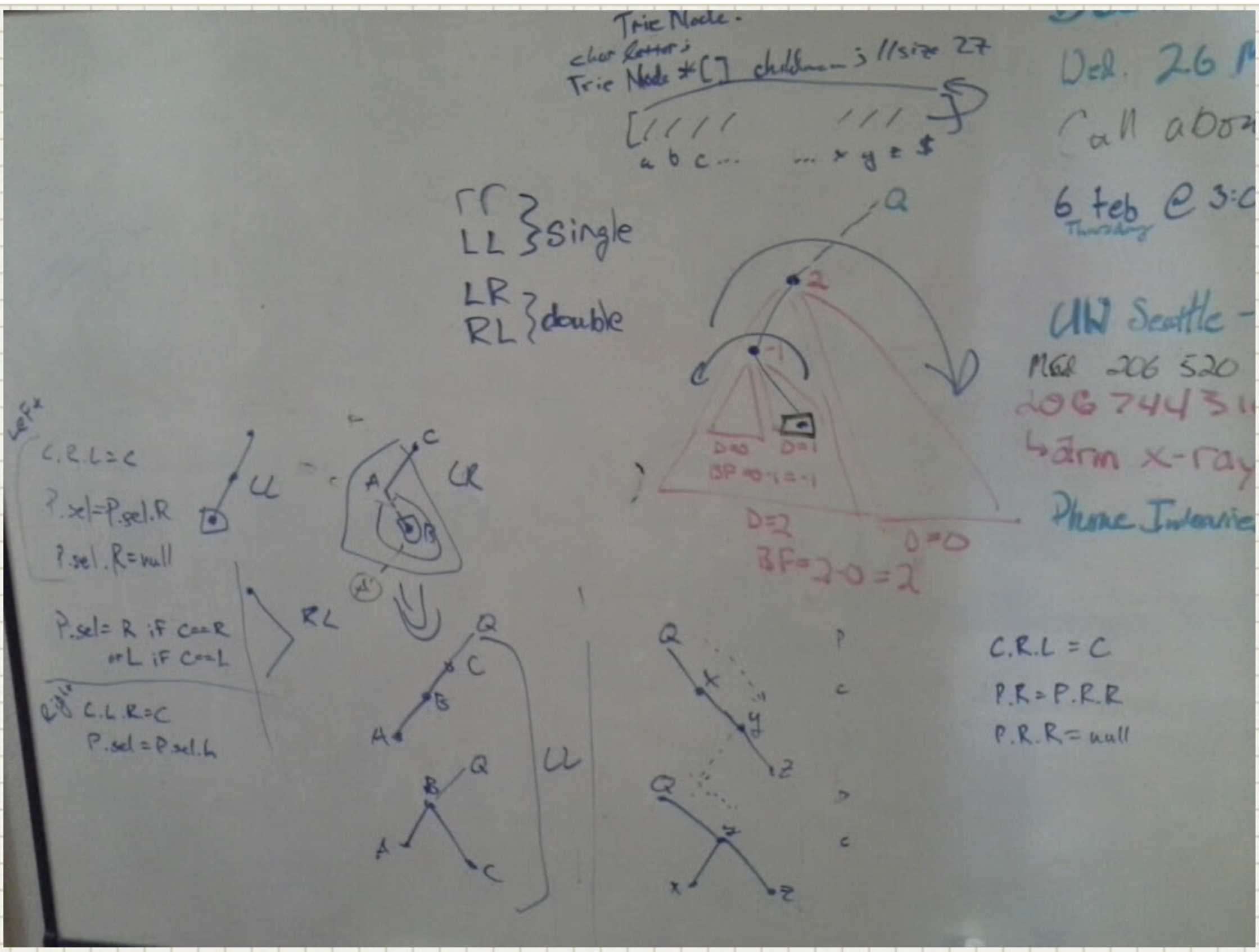
- base class ↔ super classes
  - ↳ derived class
  - ↳ subclasses



Open-closed principle

open for extension

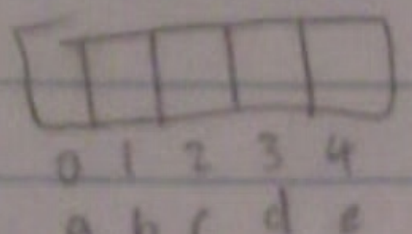
closed for modification


Trie Node.
Trie Node *[] children; //size 27
a b c ... ... x y z $
Single
LL
LR
RL
double
a
2
-1
D=0
D=1
BF=0-1=-1
D=2
D=0
BF=2-0=2
C.R.L=C
P.x=P.sel.R
P.sel.R=null
P.sel= R if C==R
or L if C==L
C.L.R=C
P.sel=P.sel.L
RL
LL
C.R.L = C
P.R = P.R.R
P.R.R = null
Wed. 26 M
Call about
6 Feb @ 3:0
Thursday
UW Seattle
MRI 206 520
206 744 3
Harm x-ray
Phone Interview

hash table - fast lookup

2/5/14

rle: char letter [E]



int index = letter - 'a'; (4)

if NULL, create new??

else, current to next pointer??

Design - class diagram, simple

Put all in .h file for Lab 4 design turn-in.

UML Notes

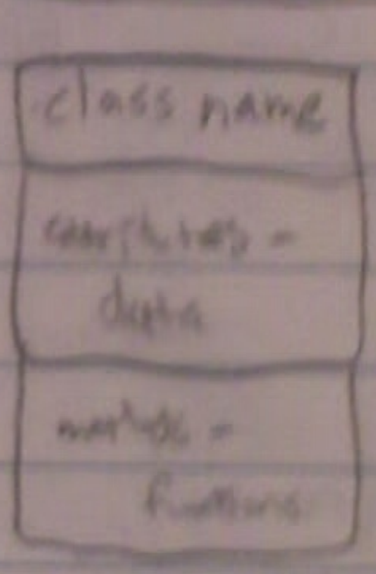
class = [ ] ex. [student]

No hand drawn UML

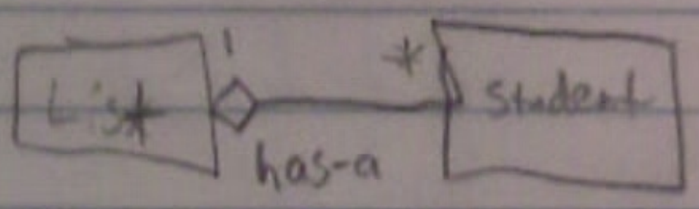
USE VIZIO for UML design

Zander only cares for composition, inheritance.

real UML (might not use in 343)

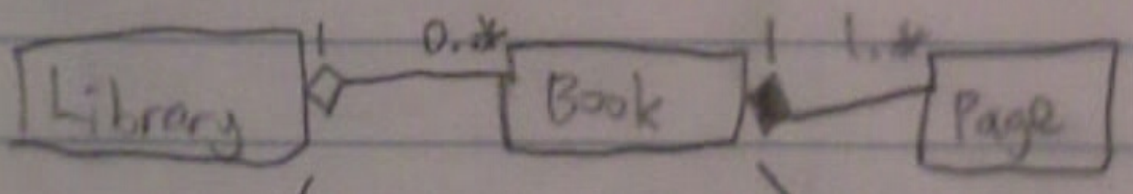


not used

ex. 

composition (aggregation in Vizio)

1 List object will have many Student objects

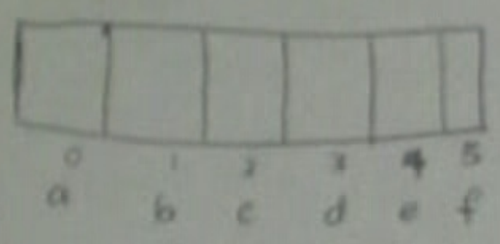
ex. 

Can exist if it doesn't have any

Can not exist if it doesn't have any.

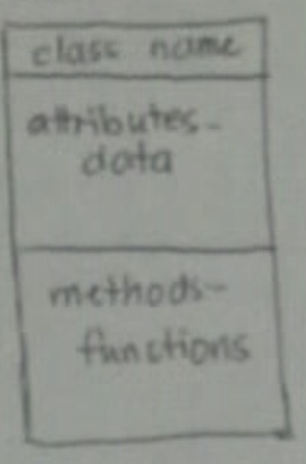
→ cont.

```
char letter, [0]
int index = letter - 'a';
```



* This is a basic Hash Table

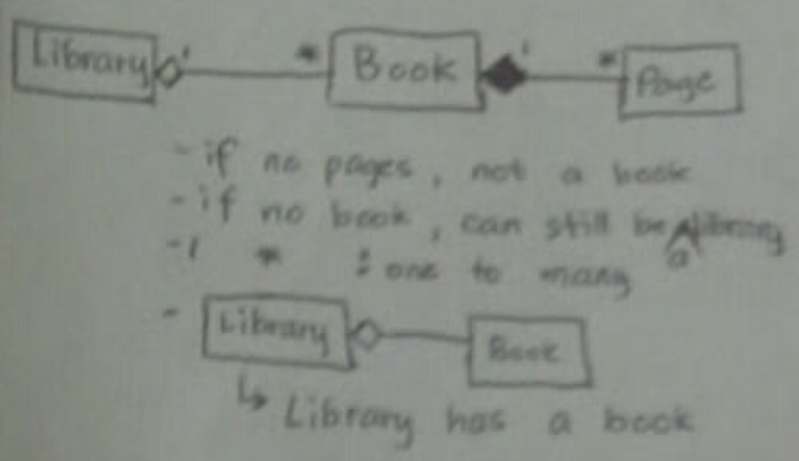
2/6/2014
CSS 343



UML

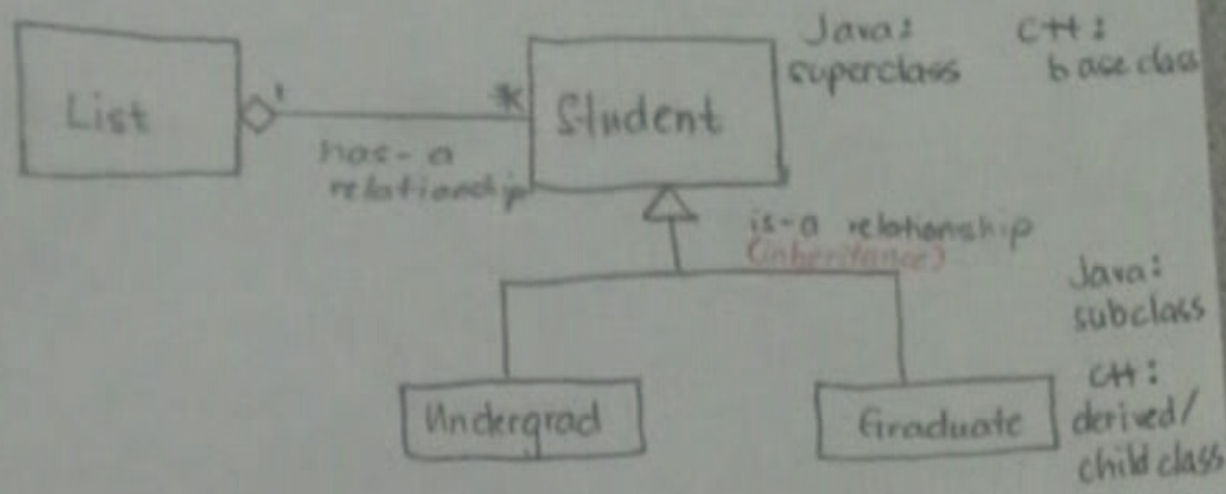
rectangle → class
diamond → composition / aggregation
- open ◇ can exist if nothing
- closed ◆ can't exist if nothing

* many
1 one
hollow arrow (△) → inheritance



- if no pages, not a book
- if no book, can still be a library
- 1 * : one to many

↳ Library has a book

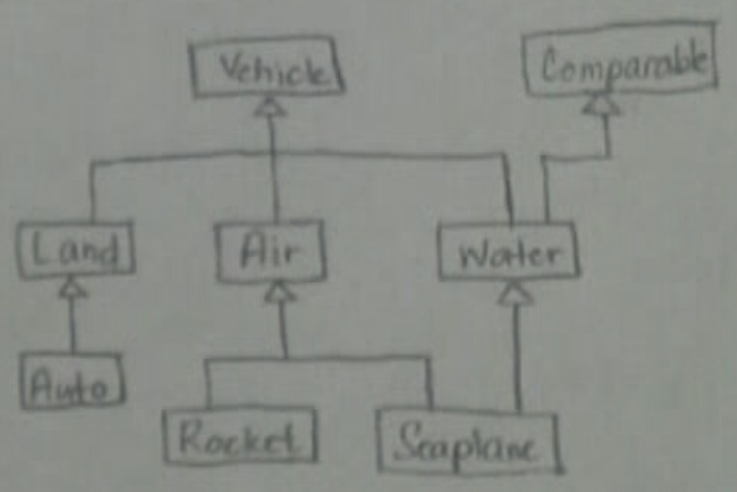


Nouns

table → manager
card
deck
player
dealer
hand
bet
pot

Verbs

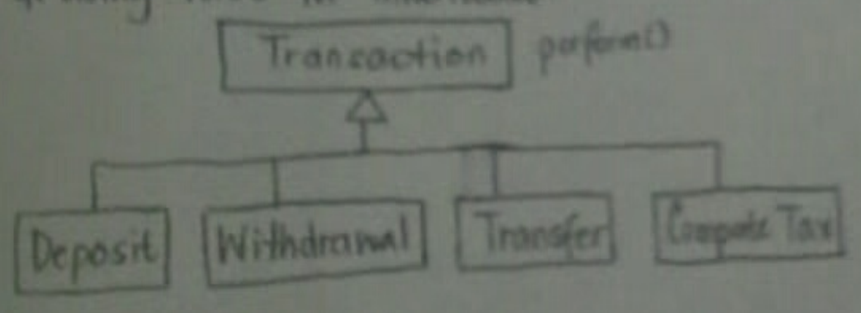
shuffle
bet/raise/call
deal
fold



Lab 5

Design

* using verbs for inheritance:



Open-Closed principle → open for extension, closed for modification

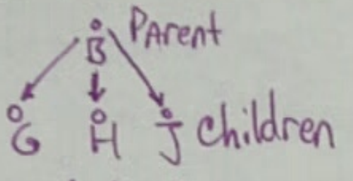
Polymorphism → doing the same tasks different ways

C - is not object-oriented
- structured language

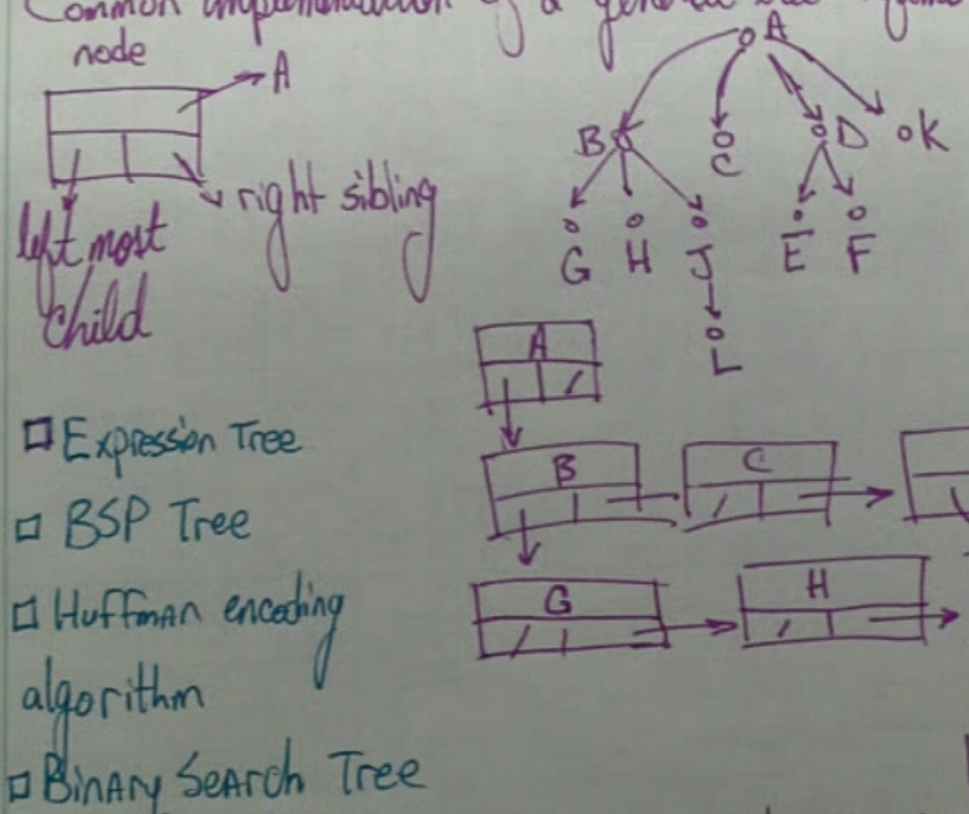
○ Bubblesort for comment how-to. What code is doing → not how.
end column comments → keep it short

1/8/2014

Trees: a single node is a tree.

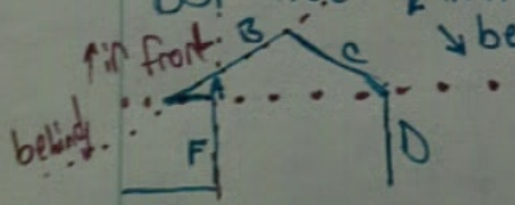


Common implementation of a general tree - leftmost child, right sibling

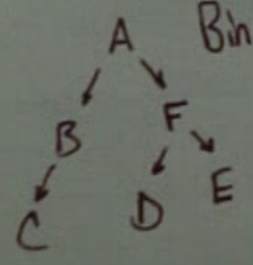


- Expression Tree
- BSP Tree
- Huffman encoding algorithm
- Binary Search Tree

BSP Tree: in front of, behind



Binary Space Partition



Encoding - Find a uniquely encoded and decoded code s.t. chars w/ higher freq have shorter encodings

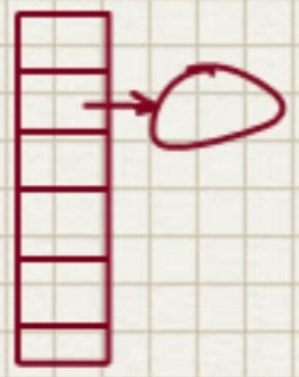
ADT Poly
data structure → Array
Binary Search Tree
- finite ordered collection

Hashing notes

N = # of items in table

B = # of buckets, # of elements in array

Open Hashing / seperate chaining



$O(1+\frac{N}{B}) = O(\frac{N}{B})$

Average # of probes when found: $1 + \sum_{i=1}^{\infty} \left(\frac{N}{B}\right)^i = \frac{N}{B-N} = \frac{1}{1-(\frac{N}{B})}$

α called load factor

sum over all i
of at least i collisions

start with fiction
get out put
move on once finished
add next content
:
add PAtron
:
Add Actions

Polymorphism

early binding - compile time

late binding → run time

Polymorphism d.vptr

vtable holds addresses of any virtual functions

every class with virtual has a vtable.

```
class B
public:
    virtual char f(){return 'B';}
    char g(){return 'B';}
    char testf(){return f();}
    char testG(){return G();}
```

```
class D : public B
public:
    virtual char f(){return 'D';}
    char g(){return 'D';}
```

```
Main
    D d;                    B
    cout<< d.testf()<<" "
    d.testG()<<endl;
cout<< d.g(); → D
```

- Changed content factory
- action factory
- gonna override display ∀ actions ∈ History.
  ↳ History HAS-A actions
- links to composition ∀ graphic
- operator overloads all pure virtual
  ↳ super class

NodeData → kinda like JAVA's object
  NO NodeData → use content as NodeData
    ↳ setData - override ∀ content
      ↳ give ioStream

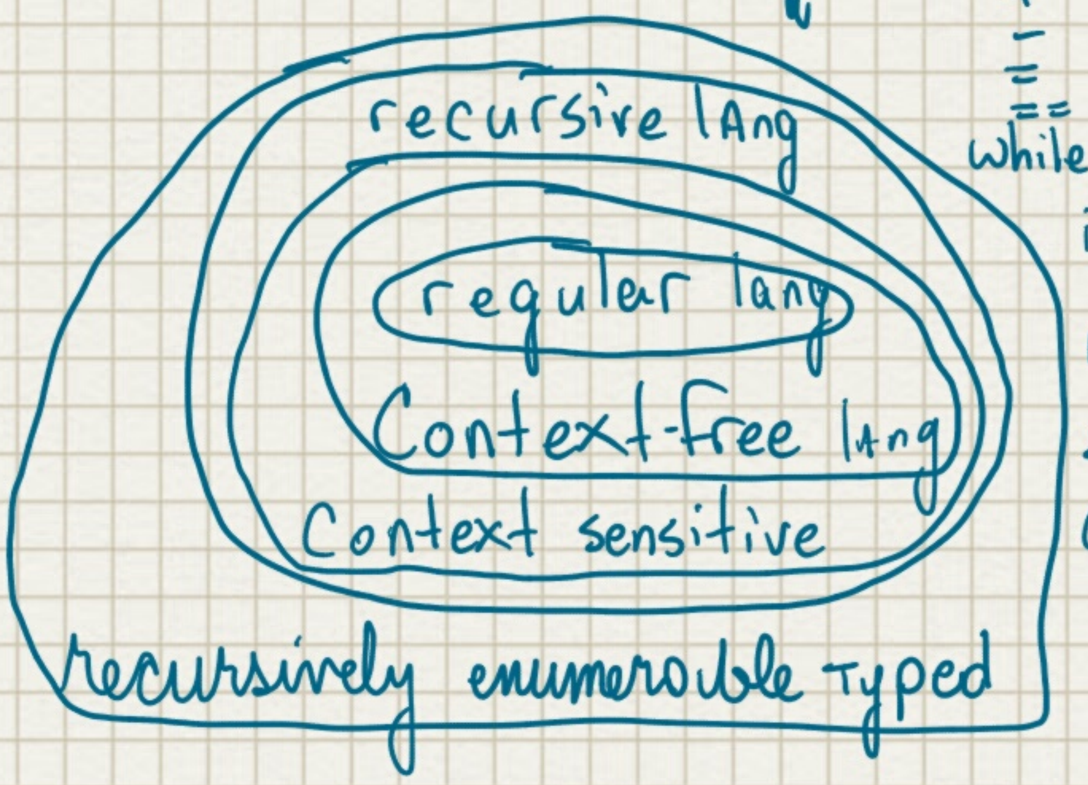
History holds actions
pass actions as ptr.

Language - a set of strings
We're interested in expressiveness of languages

Compilers

Stream of chars
↓
lexical analyzer
↓
tokens
↓
syntactical semantic analyzer
"Parser"

Chomsky Hierarchy



Tokens
(
+
-
==
while
identifier: book
number: 50, 100.27
string literal "..."
char

even a's and b's
$(aa|bb|(ab|ba)(aa|ba)^*(ab|ba))^*$

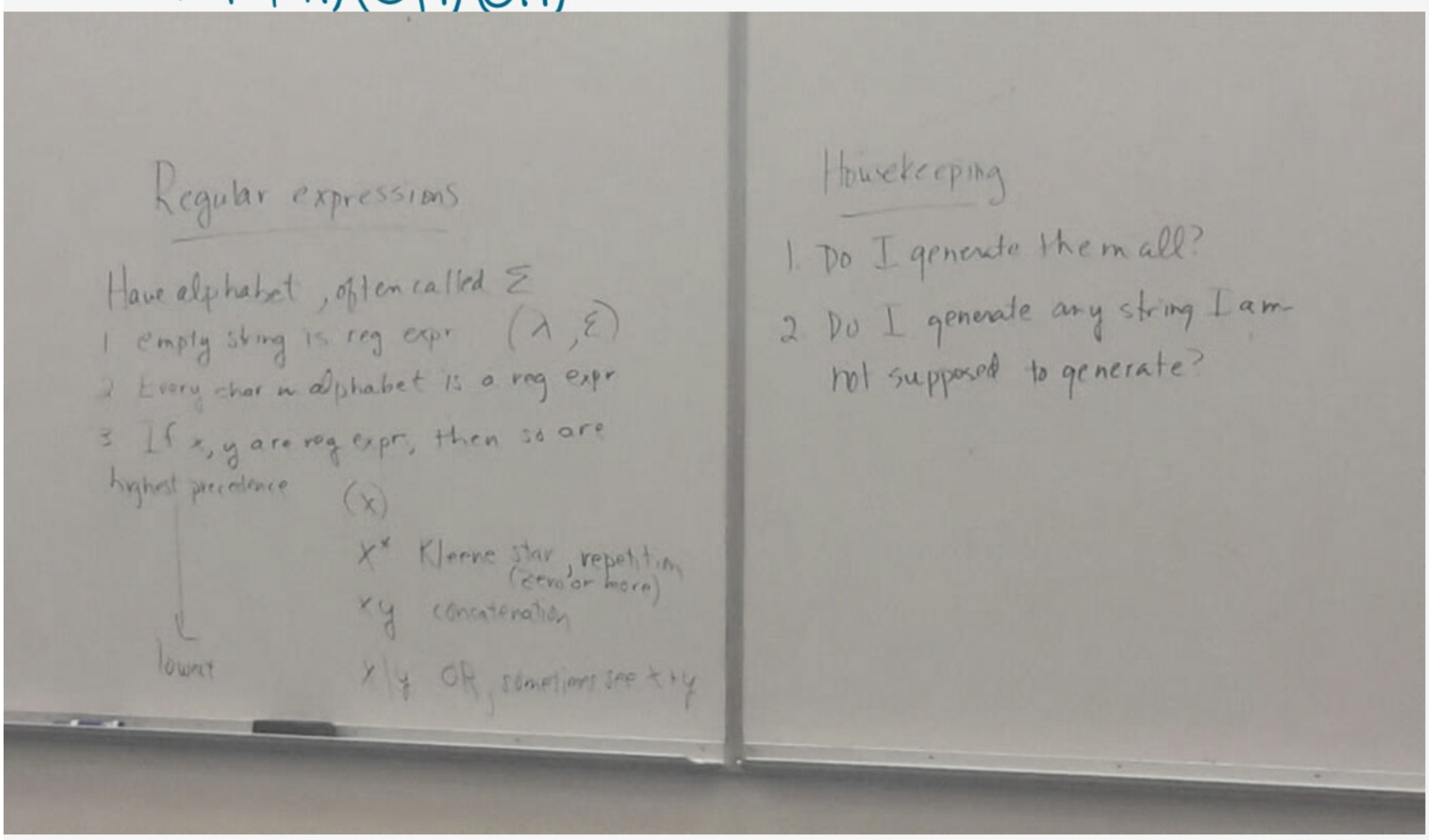
even a's
$(ab^*a|b)^*$

odd a's $(ab^*a|b)^*ab^*$

$\Sigma = \{0, 1, +, -\}$

$L_1 = \{S \mid S \text{ is an unsigned binary number}\}$

$(+|-|\lambda)(0|1)(0|1)^*$



$L_1 = \{S \mid S = na+mb,\ n,m \geq 0\}$ "a^*b^*

$L_2 = \{s \mid na+nb,\ n \geq 0\}$

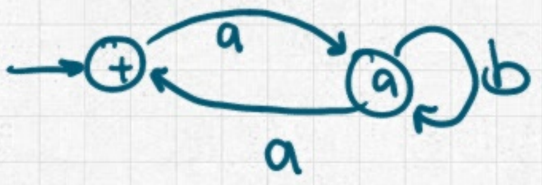
$\Sigma = \{a, b, 1, 2, \_\}$ C++ like identifier
1. must start with letter, or underscore
2. Then any letters, digits, underscores

$\Big\}\ (a|b|\_)(a|b|1|2|\_)^*$

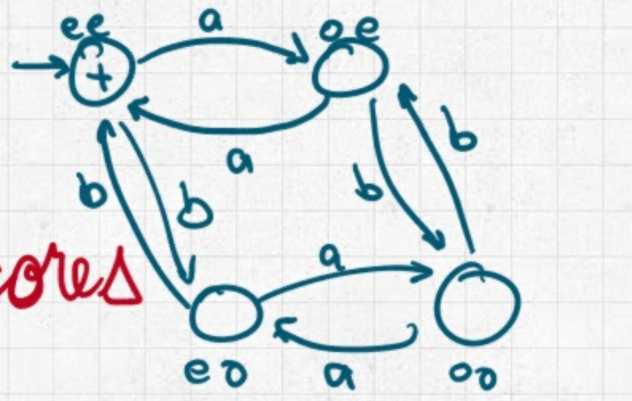
Ada-like identifier
1. must start with letter
2. Then letters, digits, underscores except cannot have two consecutive underscores

Nope $(a|b)((a|b|1|2|\_)(a|b|1|2|\_))^*$

$\Sigma = \{a,b\}$ Even a's



$\|$ Even a's, Even b's

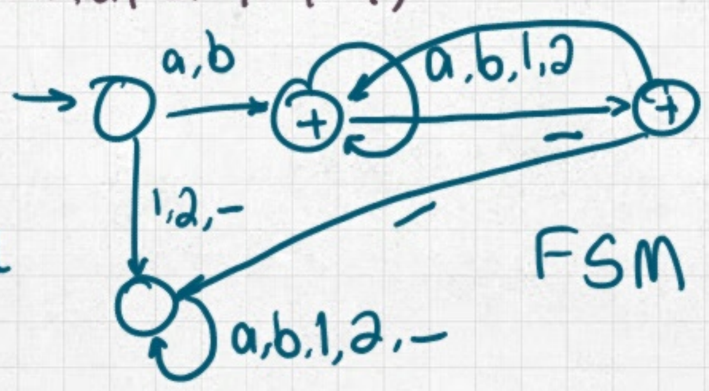


No $(a|b)((a|b|1|2|\_)(a|b|1|2))^*$

Nope $(a|b)((a|b|1|2|\_)(a|b|1|2)^*)\ (a|b|1|2|\_)$

Still no $(a|b)((\lambda|a|b|1|2|\_)(\lambda|a|b|1|2) \mid (\lambda|a|b|1|2)(\lambda|a|b|1|2|\_))^*$

Yes $(a|b)(a|b|1|2|\_(a|b|1|2))^*(\_|\lambda)$



Finite State Machine ↔ Finite Automata

Deterministic Finite State Machine

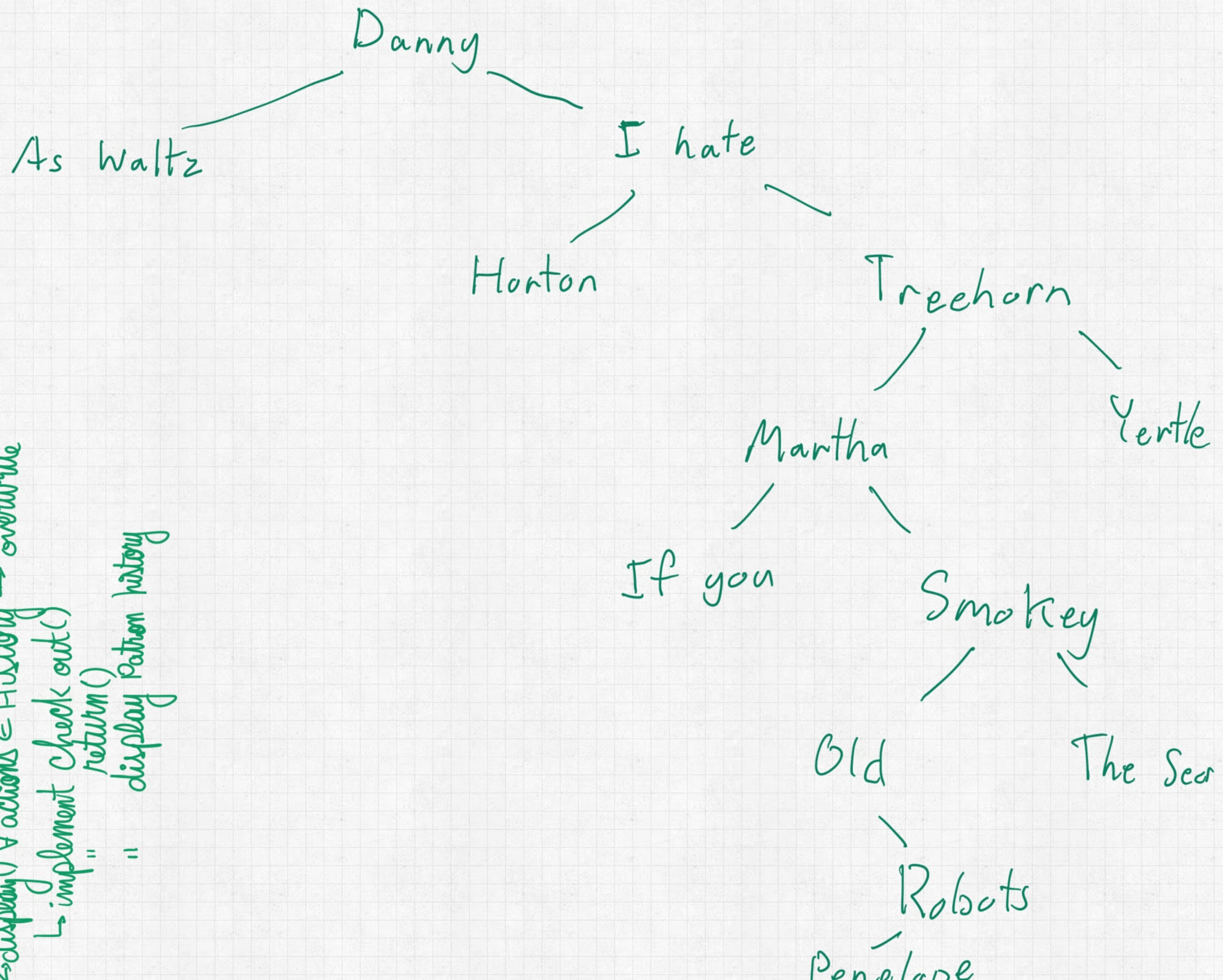
Kleeney's Theorem ∀ Regular Language
1. recognized by RegEx
2. FSM recognizes
   i. set of states
   ii. start state $\ominus \rightarrow \bigcirc$
   iii. final state $\oplus \rightarrow \textcircled{\bigcirc}$
   iv. alphabet - input
   v. transition function

Milan's thoughts
- finish display of each type of c
- Write display library
  - columns of data → aligned
- History has many actions
  ↳ display() ∀ actions ∈ History → overwrit
  ↳ implement check out()
    " return()
    " display patron history

To Do: Abstract out content factory from content.

Stretch Goals
Format names
Format headers
A.V.L. our trees

Youth



lexical analysis → tokens → parser

YACC
Bison

Context-free Grammar — Write BNF

1. set of terminals (characters, tokens)
2. set of non-terminals
3. starting non-terminal
4. set of production rules → Looks like

one nonterminal → terminal and nonterminal

⟨sentence⟩ → ⟨noun-phrase⟩⟨verb⟩⟨noun-phrase⟩

P⟨noun-phrase⟩ → ⟨noun⟩ equivalent ⟷ [⟨noun-phrase⟩ → ⟨noun⟩ | ⟨noun-phrase⟩ → ⟨adj⟩⟨n-p⟩]

P⟨noun-phrase⟩ → ⟨adj⟩⟨noun-phrase⟩

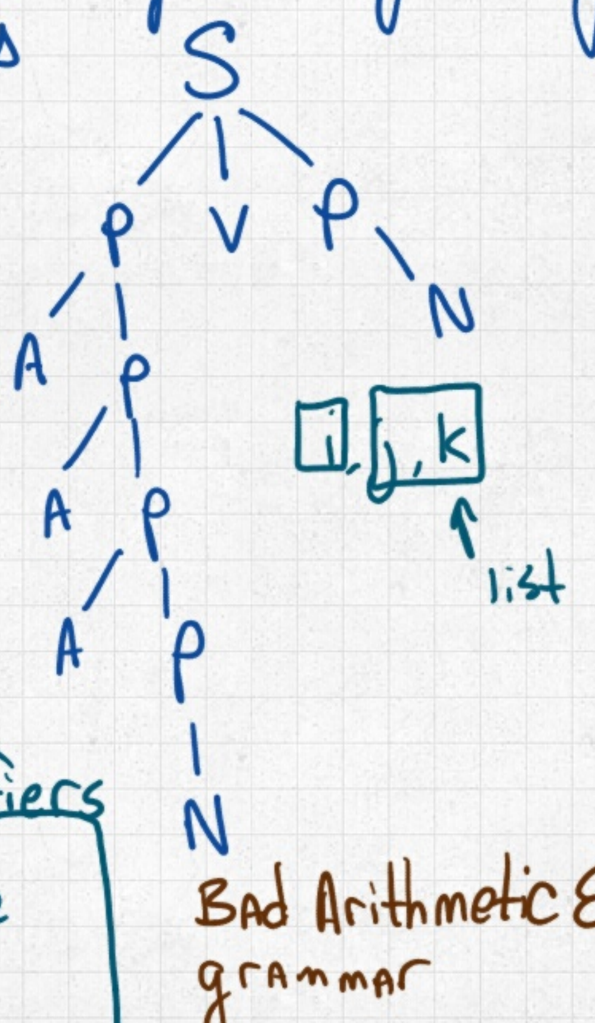
N⟨noun⟩ → cat | dog | fish | apple | ball

V⟨verb⟩ → eat | pet | juggle

A⟨adj⟩ tasty | brilliant | red

Derivation
1. S ⇒ PVP
⇒ APVP
⇒ AAPVP
⇒ AAAPVP
⇒ AAANVP
⇒ AAANVN

Demonstrate generating string
2. Parse Trees



Bad Arithmetic Expression grammar

⟨expr⟩ → ⟨expr⟩ + ⟨expr⟩ |
⟨expr⟩ - ⟨expr⟩

E → E+E | E-E | E*E | E/E |
(E) | ⟪number⟫ | ⟪ident⟫

E is ambiguous ⟷ ∃ > 1 distinct parse tree for 1 same string!


```
int i,j,k;
float x,y;
char a,b,c;
bool done;
```

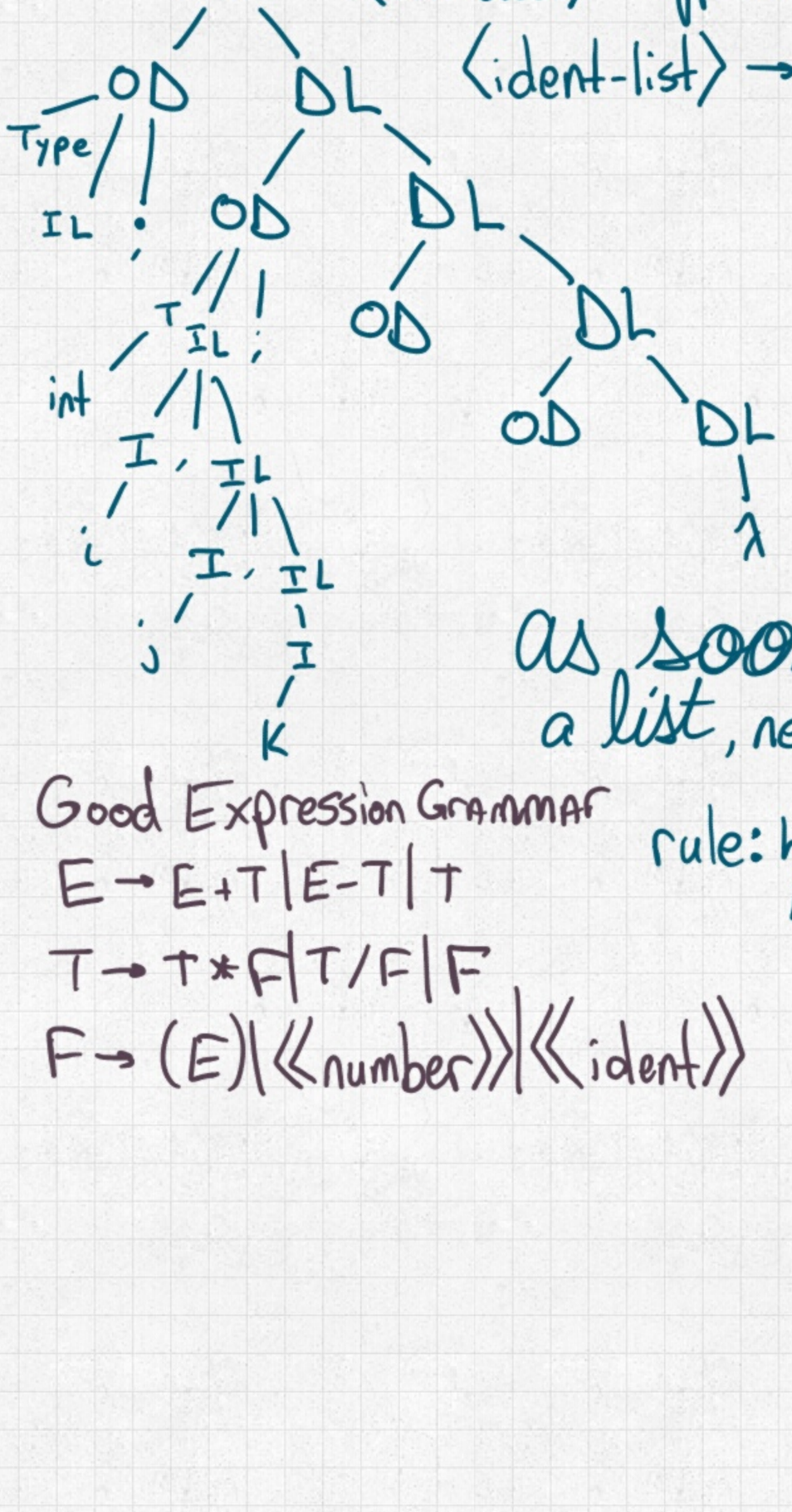
⟨decl-list⟩ → λ

List make nonterminal

⟨decl-list⟩ → λ | ⟨one-decl⟩ ⟨decl-list⟩

DL ⟨one-decl⟩ → ⟨type⟩⟨ident-list⟩;

⟨ident-list⟩ → ⟪identifier⟫



Identifiers

```
while
>
<
=
==
(
)
.
.
.
identifier
number
literal string
```

as soon as you have a list, need a new non-terminal

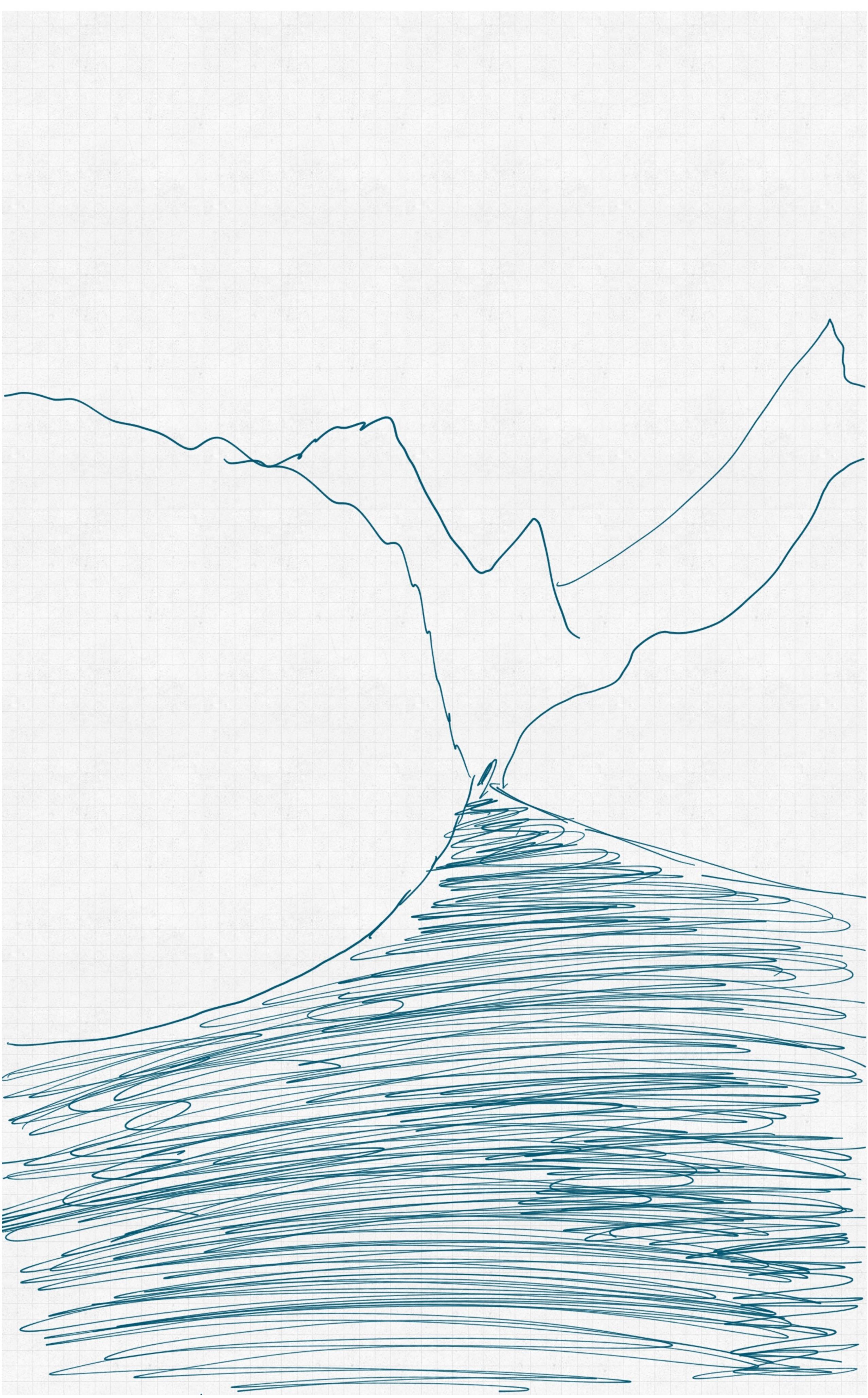
rule: haz identifier
↳ comma, ident-list

Good Expression Grammar

E → E+T | E-T | T

T → T*F | T/F | F

F → (E) | ⟪number⟫ | ⟪ident⟫



Semi-comprehensive

↳ Focus on second half

↳ 8×11 cheat sheet, Both sides

↳ till 10:30

Hashing → linear/Quadratic Probing

Complexities

inheritance/polymorphism → design

Open/Close principle

↘ code: read/write (Polymorphism)

at least one (2?) focused on Poly/inherit

implementation under the hood

Regular Expression → Languages → Turing Machines (staff about it)

↳ "Put it all together"

↳ Factory Pattern/Command Parent

Transaction

Checkout return