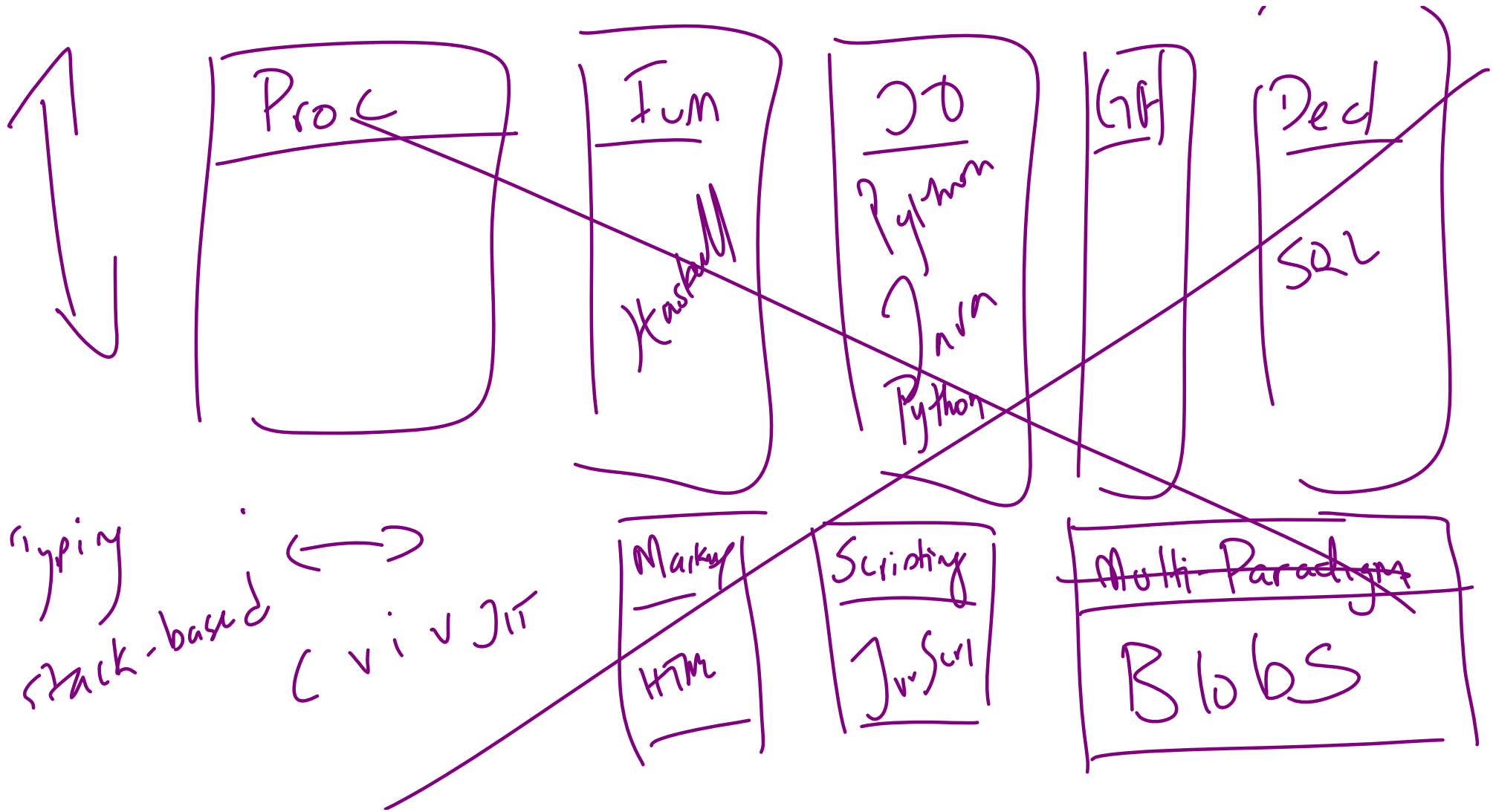
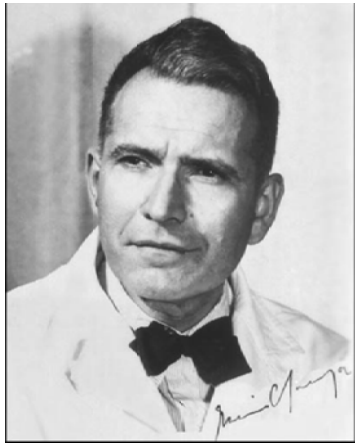
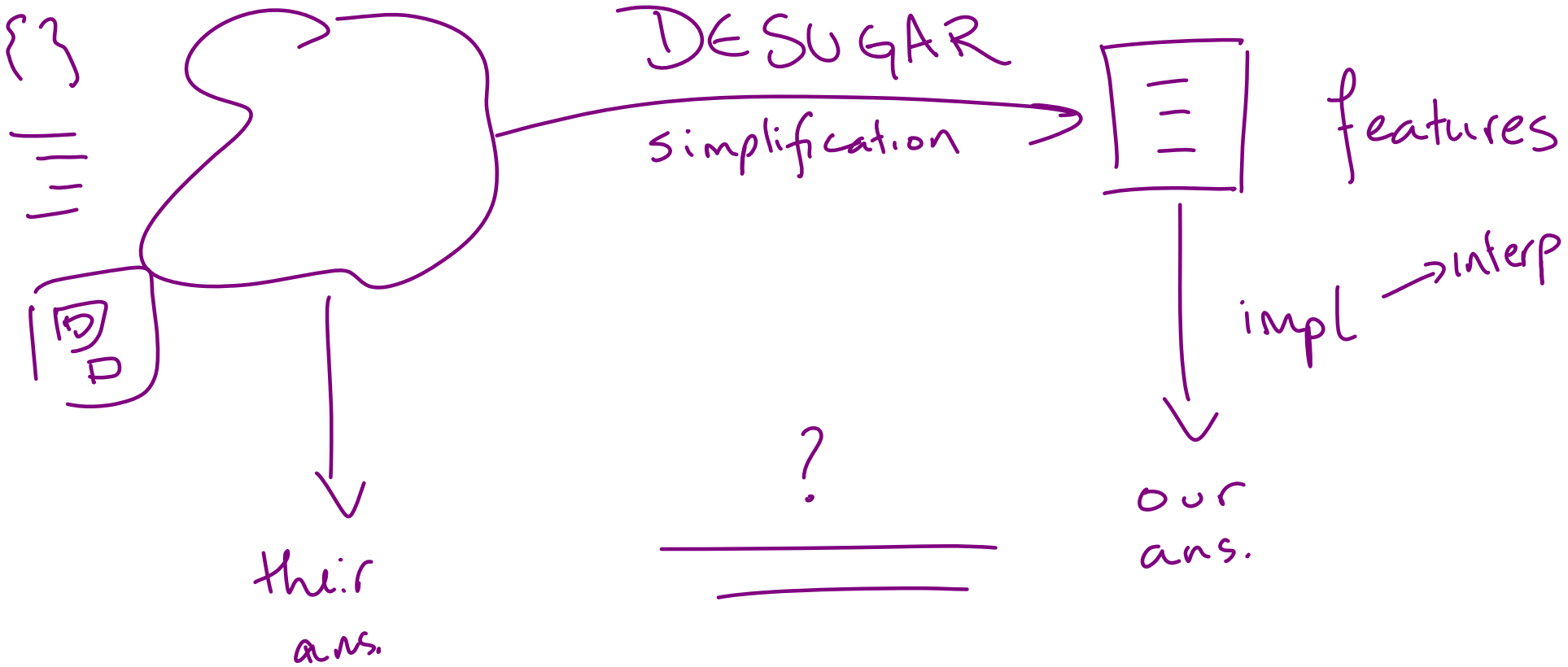
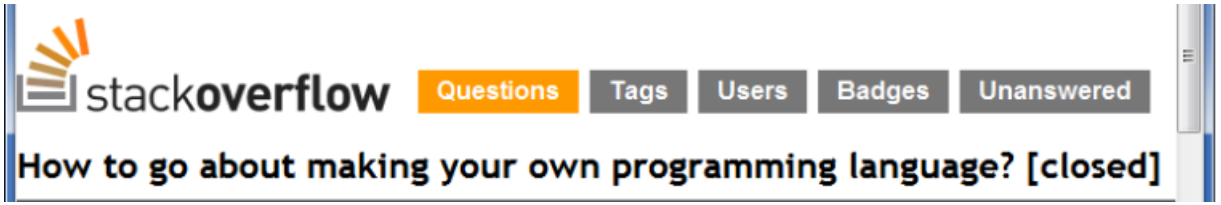

HOW?


Proc
Fun
Haskell
OO
Python
Java
Python
GPL
Decl
SQL
typing
stack-based
<->
C
V8
JIT
Markup
HTML
Scripting
JavaScript
Multi-Paradigm
Blobs



Credits:
- en.wikipedia.org
- gap.entclub.org
- history.nih.gov


DESUGAR
simplification
features
impl
interp
?
their ans.
our ans.



5

I'd say that before you begin you might want to take a look at the Dragon Book and/or Programming Language Pragmatics. That will ground you in the theory of programming languages. The books cover compilation, and interpretation, and will enable you to build all the tools that would be needed to make a basic programming language.

I don't know how much assembly language you know, but unless you're rather comfortable with some dialect of assembly language programming I'd advise you against trying to write a compiler that compiles down to assembly code, as it's quite a bit of a challenge. You mentioned earlier that you're familiar wtih both C and C++, so perhaps you can write a compiler that compiles down to C or C++ and then use gcc/g++ or any other C/C++ compiler to convert the code to a native executable. This is what the Vala programming language does (it converts Vala syntax to C code that uses the GObject library).

As for what you can use to write the compiler, you have a lot of options. You could write it by hand in C or C++, or in order to simplify development you could use a higher level language so that you can focus on the writing of the compiler more than the memory allocations and the such that are needed for working with strings in C.

You could simply generate the grammars and have Flex and Bison generate the parser and lexical analyser. This is really useful as it allows you to do iterative development to quickly work on getting a working compiler.

Another option you have is to use ANTLR to generate your parser, the advantage to this is that you get lots of target languages that ANTLR can compile to. I've never used this but I've heard a lot about it.

Furthermore if you'd like a better grounding on the models that are used so frequently in programming language compiler/scanner/parser construction you should get a book on the Models of Computation. I'd recommend Introduction to the Theory of Computation.

4

Take a look at ANTLR. It is an awesome compiler-compiler the stuff you use to build a parser for a language.

Building a language is basically about defining a grammar and adding production rules to this grammar. Doing that by hand is not trivial, but a good compiler-compiler will help you a lot.

You might also want to have a look at the classic "Dragon Book" (a book about compilers that features a knight slaying a dragon on the front page). (Google it).

Building domain specific languages is a useful skill to master. Domain specific languages is typically not full featured programming language, but typically business rules formulated in a custom made language tailor made for the project. Have a look at that topic too.

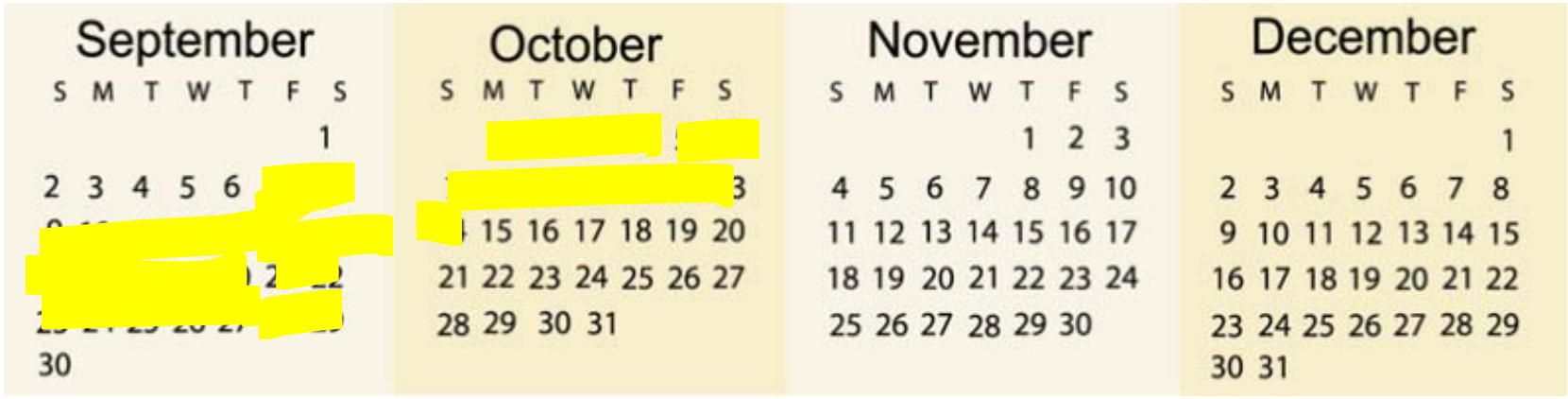
- Collaboration and honesty policy

- Subscribe on Piazza
  PL Online / PL 1 / "jiveyleague"
- Remember your course code (from Joe)

- This course is on-line!

- Grade caps

- Course pace (brutal)
- First homework out FRIDAY, due in a week
- Learn Racket! -- Piazza "Class



September

S	M	T	W	T	F	S
						1
2	3	4	5	6		
30						

October

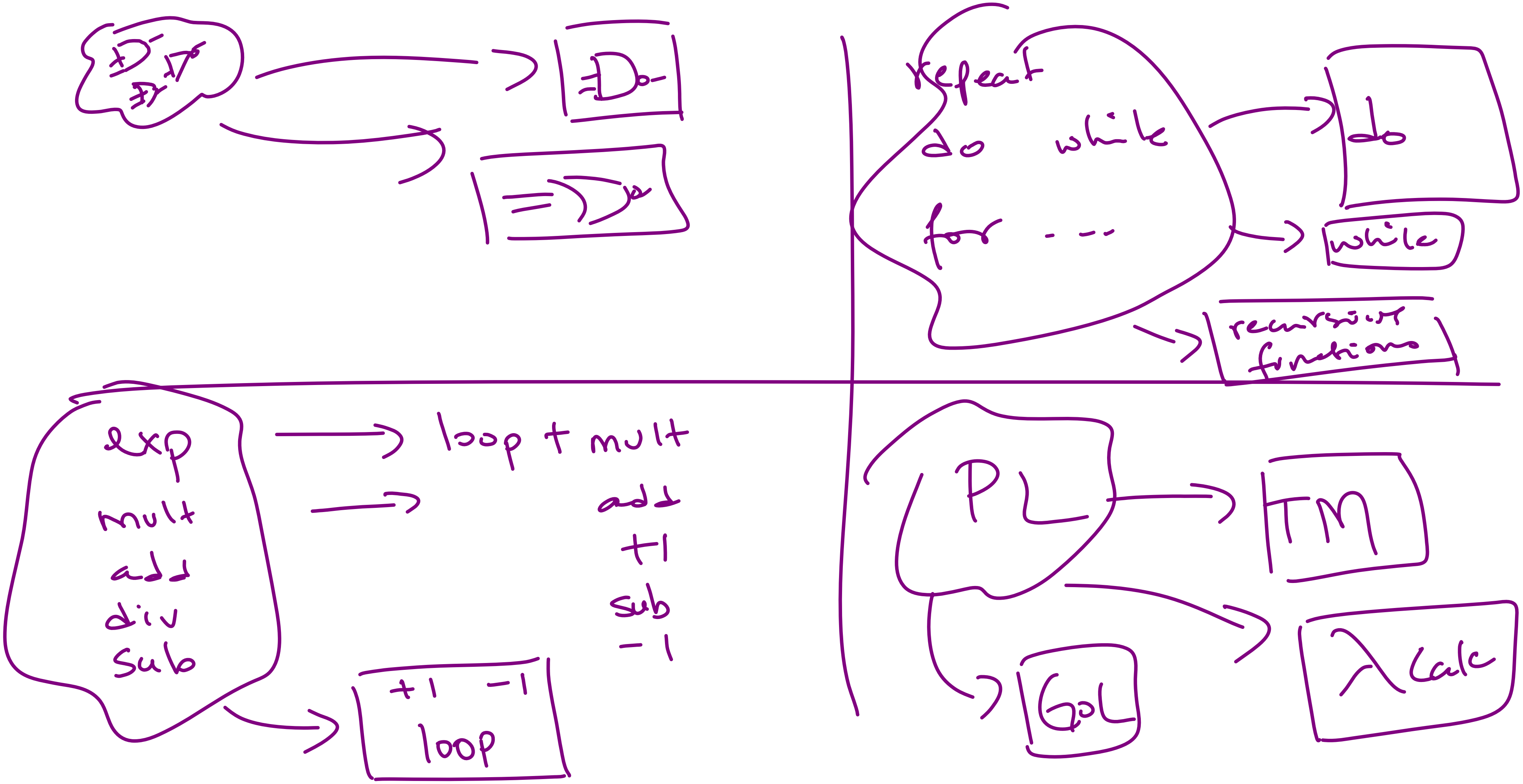
S	M	T	W	T	F	S
	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

November

S	M	T	W	T	F	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

December

S	M	T	W	T	F	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					


repeat
do
while
for ...
do
while
recursive functions
exp
mult
add
div
sub
loop + mult
add
+1
sub
-1
+1
-1
loop
PL
TM
λ calc
GoL