Summary:

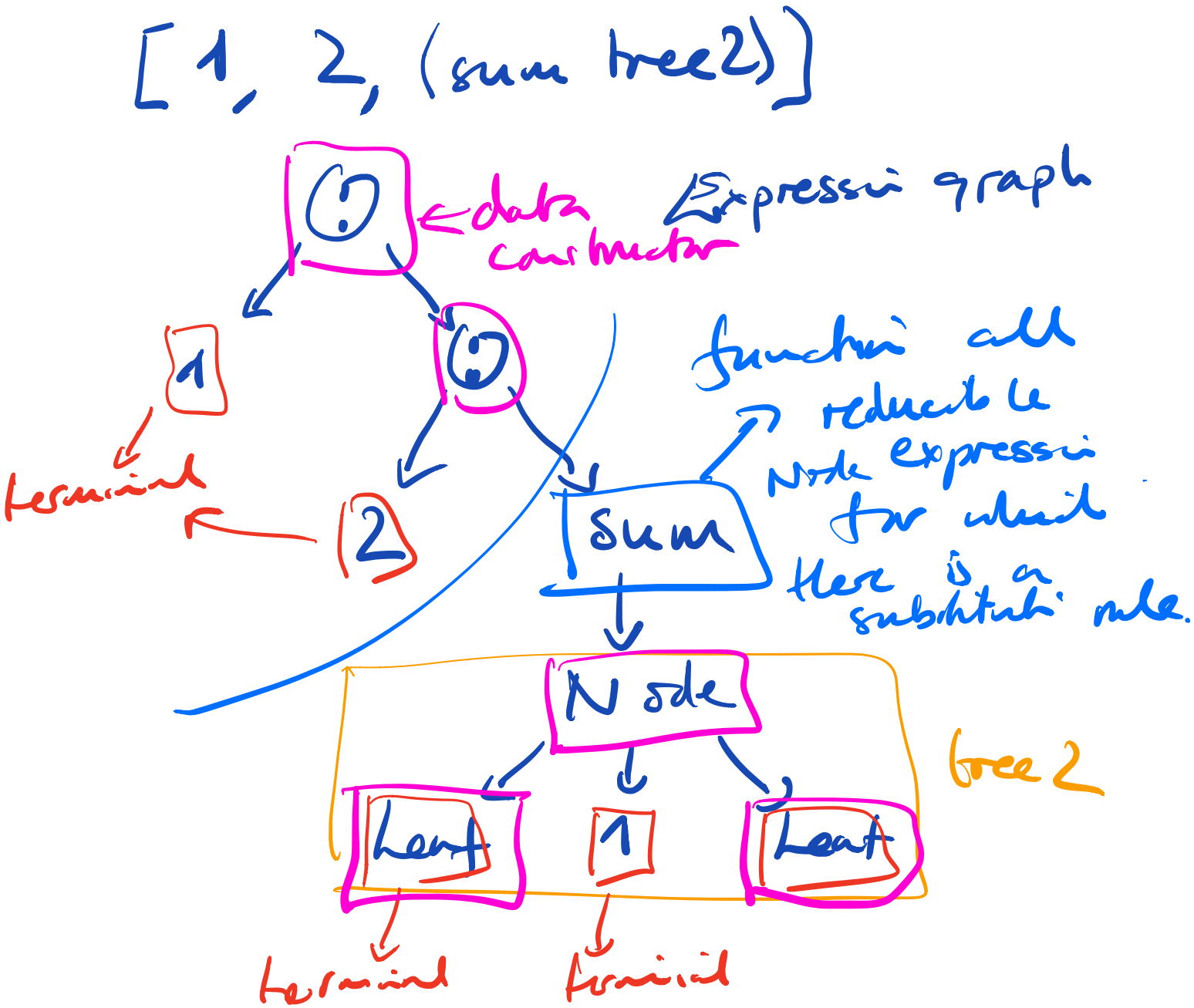
- Mathematically structured programming
- Types types types
- Think about interfaces / API boundaries

Reducible expressions

```
data Tree a = Leaf
            | Node (Tree a) a (Tree a)

tree = Leaf
tree1 = Node Leaf 1 Leaf

sum :: Num a => Tree a -> a
sum Leaf = 0
sum (Node l x r) = x + (sum l)
                     + (sum r
```


[1, 2, (sum tree2)]
(:)
data constructor
Expression graph
1
(:)
terminal
2
terminal
sum
function call
reducible
Node expression
for which
there is a
substitution rule.
Node
Leaf
1
Leaf
tree 2
terminal
terminal

Cryptographic libraries
→ often written in C.
⇒ want control on execution.

∴ memory unsafe.

∴ type system is weak.

New crypto stuff ⇒ Rust.

in C:

```
fd = fopen(...);
read(fd, 1, &array);
```

```
foo = malloc(array, ...)
fclose(fd);
read(fd, n, ...);
```

Rich type system encode: that this error is illegal at compile time.

session types or "typestate pattern".

Exam:

→ look at past papers.

This year: open book.

→ No bookwork questions.

→ Short programming questions.
"write some code to do X".

4 questions: (2 (mostly) Haskell
2 (~) Java)

60 marks in total (2 questions)
> programming examples each 20
"explain this"

40 marks (each 20 marks)
→ more "short essay"
synthesis of ideas/concepts
There don't really exist in past papers.

→ So marked paper with have
one of these, explanations

Paper is nominally 2hrs.
→ This year you get 24hr period to do it.