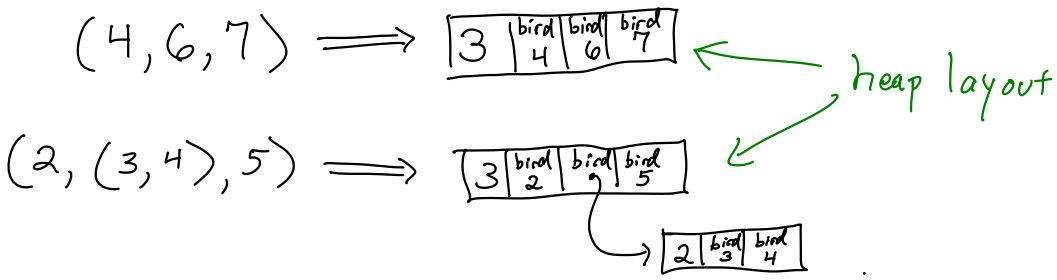
Eagle

* Tuples, indexing, istuple
* Representation: pointer to heap memory
  - all bird pointers end in '01' but point to an address divisible by 8
  - pointer points to heap with contents: [size | el | ... | el]

binary representation



"Functional languages" ← I don't like this term.

- First-class functions: functions are values
- Partial application: you can "call" a function with some of its arguments
- Anonymous functions: functions don't need names

```
let op = compile_expression env in
List.map op expr_list
```

```
List.map (fun n -> n * 2) [4; 6; 8]
```

	OCaml	Rust	Python	Javascript	Ruby	C++	C	Falcon
First-class functions	✓	✓?	✓	✓	✓	11?	✗	✓
Partial application	✓	✓?	✗	✗	✗	✗	✗	✓
Anonymous functions	✓	✓?	✓	✓	✓	11?	✗	✗

Falcon

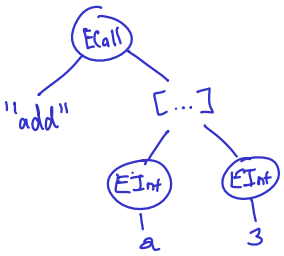
separated by spaces, no parens

```
<decl> ::= def <ident> <param_list> = <expr> end
<expr> ::= ..... ← no ECall
         | <expr> <expr>
```

Syntax

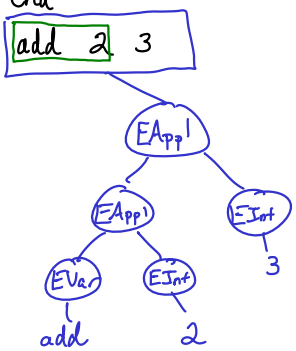
Dove

```
def add(x,y)
  x+y
end
add(2,3)
```



Falcon

```
def add x y =
  x + y
end
add 2 3
```



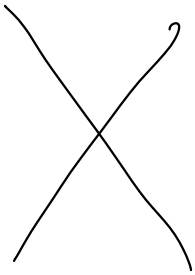
Semantics

Dove

- create a fn named "add"
- call add with 2 and 3

Falcon

- create fn called "add"
- pass 2 to the add function to produce a function waiting for y
- now pass 3 to the result



```
def add x y =
  x+y
end
let inc = add 1 in
inc 5
```

What does "add 1" produce?

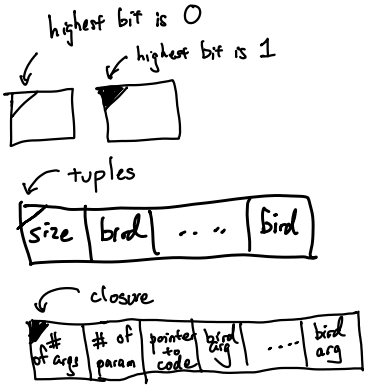
What do I need to remember?

- which function?
- # of args fn wants
- # of params I have
- I have a "1"

↓

Pointer to heap memory

Heap contains



add 1