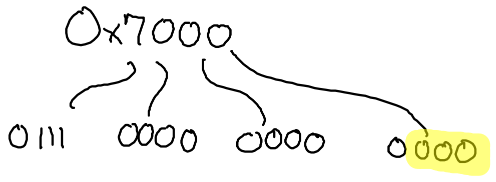
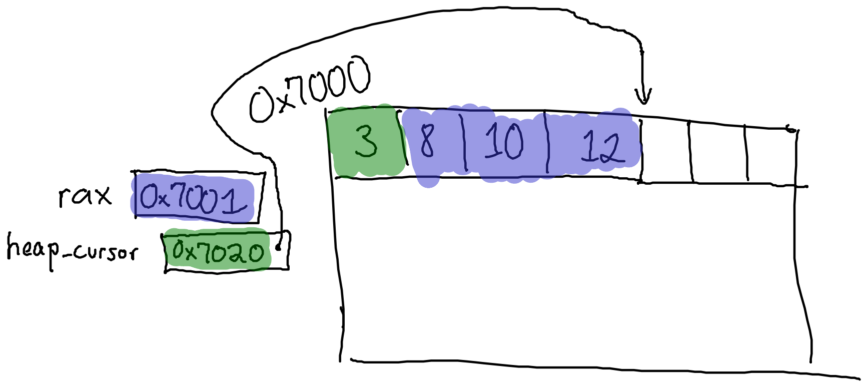
Reminder : Eagle

Add tuples (pairs, triples, ...)

<expr> ::= .... | (<expr>, <expr>, ...)
| <expr>[<expr>]



let a = (4, 5, 6) in
...



Falcon

"functional language"

```
List.map (fun a -> a+1)
list (map (lambda a: a+1, ___))
```

- Partial application: can apply a fn to some (not all) of args and get back a fn waiting for rest
- Anonymous function: can write fns using "literal syntax" rather than having to name them
- First-class functions: functions are values

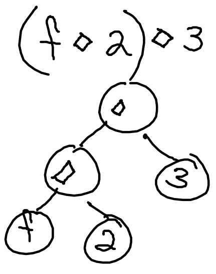
language / feature	OCaml	Python	Matlab	Lua	C	Java	C++	Falcon
~~Partial~~ Partial Application	✓	×	!!?	×	×	×	×	✓
Anonymous functions	✓	✓	!!?	✓	×	8?	11?	×
First-class functions	✓	✓	!!?	✓	× (function pointers don't count)	8?	11?	✓

Falcon

1 Syntax

<decl> ::= def <ident> <param> ..... = <expr> end

$\langle expr\rangle ::= \cdots \mid \underbrace{\langle expr\rangle\ \langle expr\rangle}_{\text{EAppl}}$



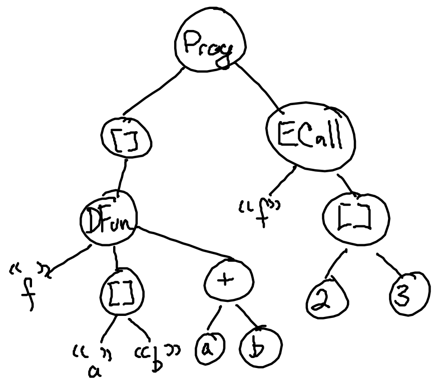
2 Semantics

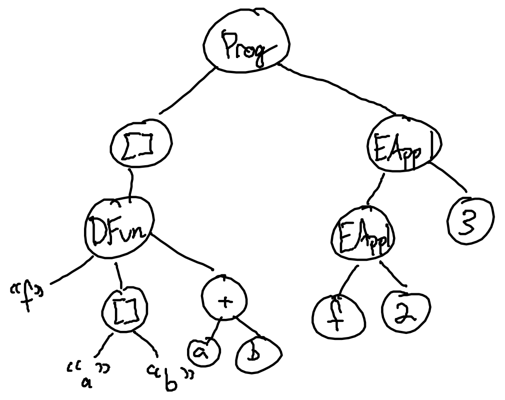
Dove

def f(a,b)
  a+b
end
f(2,3)

⇓

5





Falcon

def f a b =
  a+b
end
f 2 3

⇓

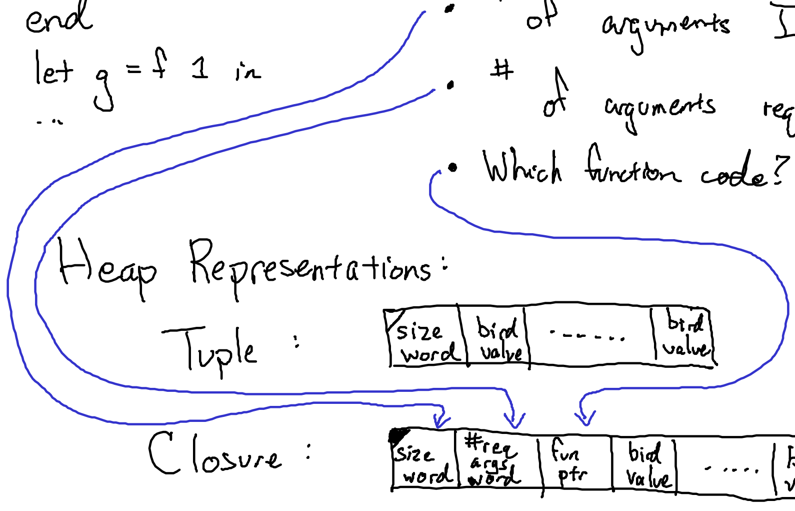
5

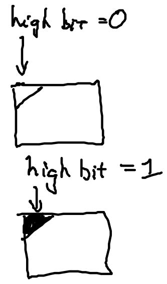
---

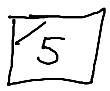
def f a b =
  a+b
end
let g = f 1 in
..

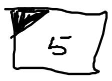
- All of arguments I have collected
- # of arguments I have collected
- # of arguments required
- Which function code?

} "closure"





 = 5

 = 0x8000000000000005