GS Lab

global

↓

I have this thing to share

extern

↓

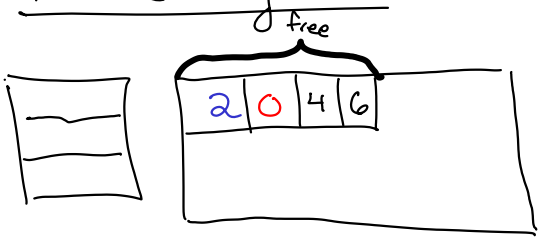
I need this thing

Garbage Collection

C++11 (but Rust is better at this)

```
unique_ptr<Foo> p = ... ;
unique_ptr<Foo> q = move(p);
```

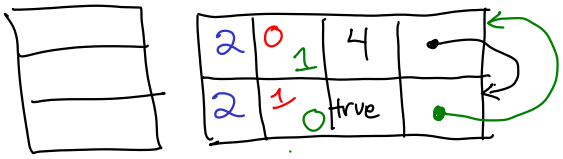
Ref. Counting




```
let x =
  let t = (2,3) in
  let u = t in
  u[1]
in
;
```

Assumptions
- No making pointers other than by copying or by allocation
- I can tell when you copy a pointer

What about cycles?



Cache
f : α → β
dict : Map<α, β>

How to solve:

C++ : programmer's problem

Swift : weak reference ← weak count / strong count

Python : occasionally search for cycles

Mark & Sweep

1. Mark every reachable obj
2. Iterate over the heap
   - if not marked, then free

Assumptions

- Somehow store mark
- Can't fabricate ptrs
- Can iterate over heap.

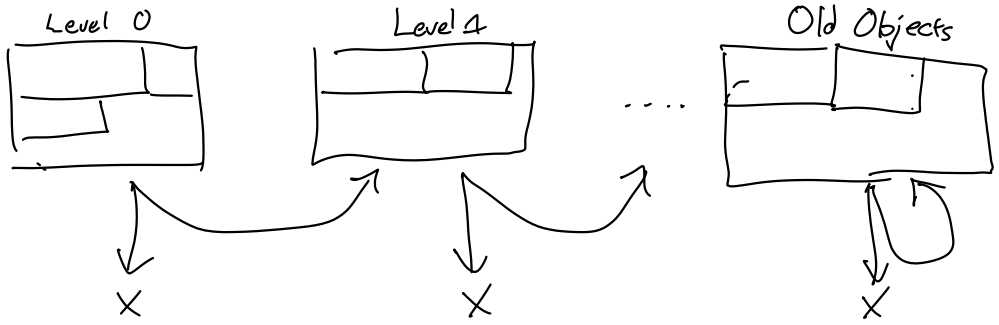
Mark & Don't Sweep

Changing interpretation of "mark"

Generational GC

Generational Hypothesis: objects which can be freed are usually new

Multiple Heap Spaces: new stuff → old stuff



"Stop-the-world" GC — bad implications any time you interact w/ the world

Incremental GC — Break up work of GC

Concurrent GC — Multiple threads

Optimistic locking