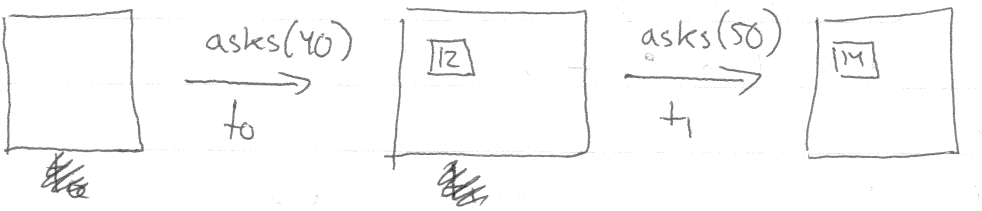
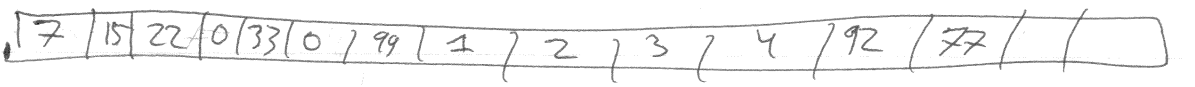
Sound, language-implementation controlled MM
└ "garbage collection" (GC)

Your program:
- Asks for memory
- Changes memory

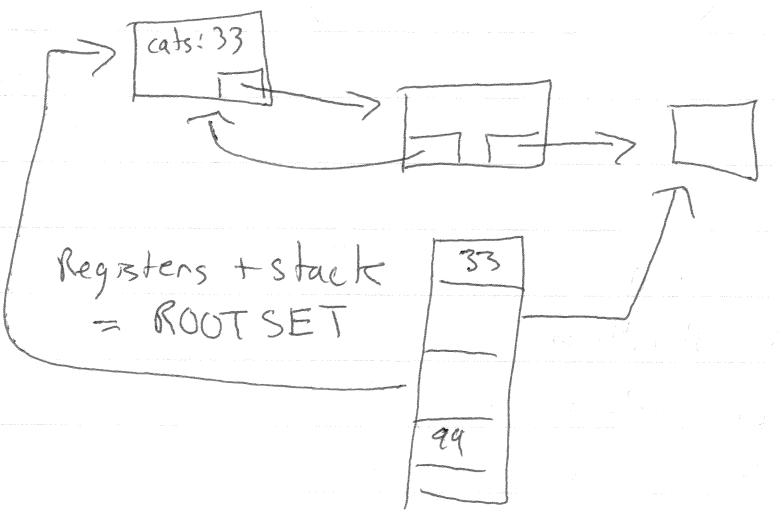
- changes to values ← don't
- changes to pointers ← matter



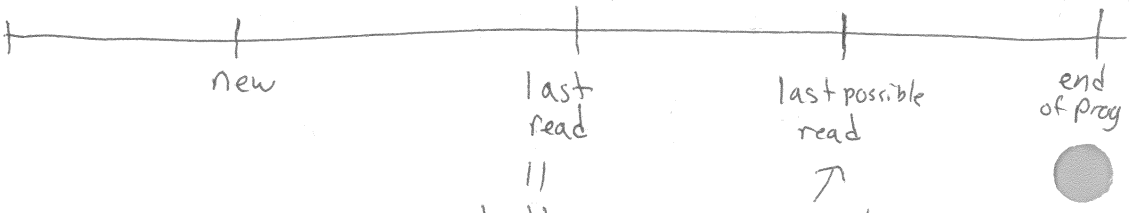
Memory Actual



Memory Abstract — Object Graph ← GC + you need it



types of mem (heap) and of root set




```
0:   x = new (23);
       ... mention x ...
99:  print(x);
       ... long computation dnm x ...   (defines y)   y = scanf
200: if (y > 29) {
201:   print(x * 2);
     }
     ~~return~~  ... big comp dnm x ...
     return
```

```
global list = ...
  x = new (23);
      ...   m x
  list->add (x);
      ...   dm x
  ret
```

scope ≠ extent

scope || when a variable is meaningful

extent || when an object is meaning || liveness

Basic G C algorithm

① Free Space = All Mem

On Req for mem x, do:

```
if(FS > x) {
    FS -= x
} else {
```

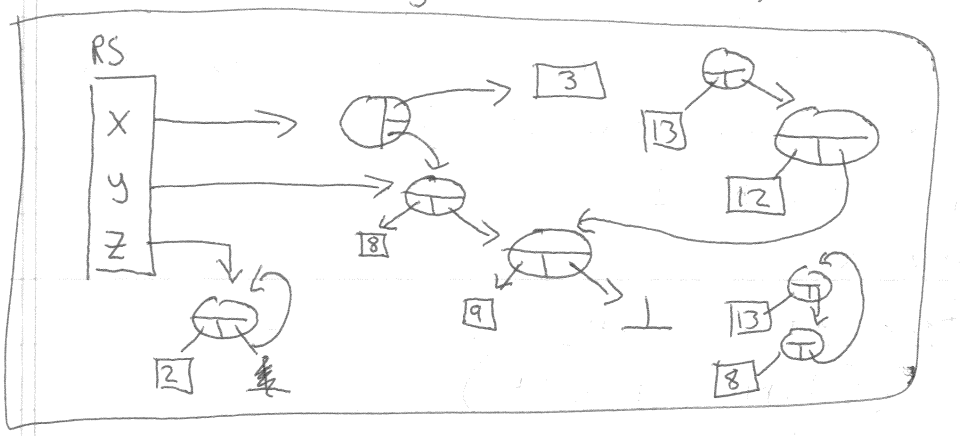
② figure out what is live (ie has a name)

③ free other stuff

repeat

```
}
```

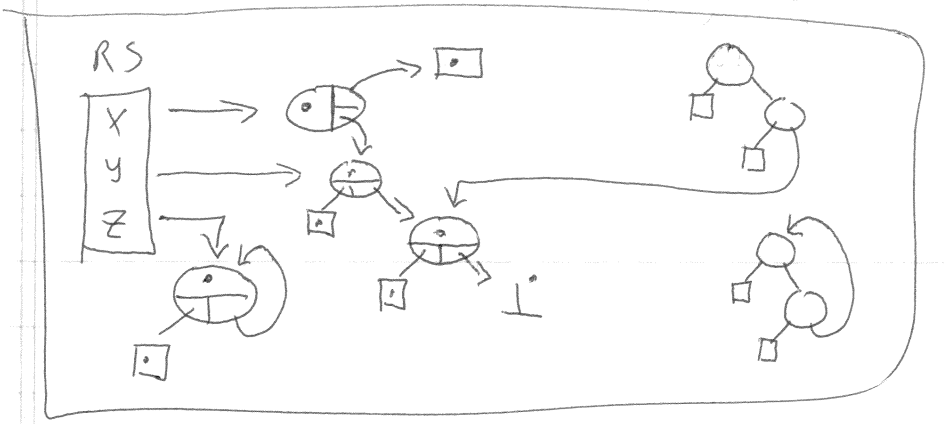
Figure out what is live (i.e. has a name)
— tracing
— computing reach-ability



live =
∃ path from root to obj

dead =
∀ path from root, dest ≠ obj

Mark + Sweep (McCarthy 1959) (1st "use" Java 1995)



mark =
follow paths from RS and put dots
O(live)
memory

~~sweep =~~

sweep = for every obj in memory,
if marked, unmark
o.w, free()

O(objs)
"live+dead"

Boehm - GC (-lgc)

malloc — O(lg n)
free — O(lgn) ~~x O(dead)~~
collect — O(lgn) x O(dead) + O(objs)
collect' — O(dead) + O(objs)
= O(objs)

11-4/ Memory Overhead (a)

overhead in time because of delayed frees
(time space trade off)

inherent overhead
- mark
- type information
  (vtable in C++)
  ptr

32-bit = OO

64-bit = O—O
         16 OO