Recitation 8: Memory management – a brief introduction

What is memory?

- Qs to class:
  - What types of memory does an executing program have and what are its purposes?
  - What is dynamic memory allocation?
  - Tell me about a language you work with and whether it supports dynamic memory allocation. If so how?

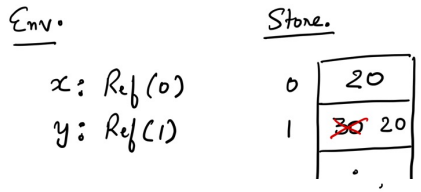
Lettuce: Explicit References

- newref(expr) → create a new cell in memory and set the initial content of the cell to whatever “expr” evaluates to.
- deref(expr) → expr evaluates to a pointer/reference to a cell in memory then fetch the contents of that cell.
- assignref(expr1, expr2) → expr1 is a reference to a cell in memory and expr2 is another value. Place the value of expr2 into the cell that expr1 points to.

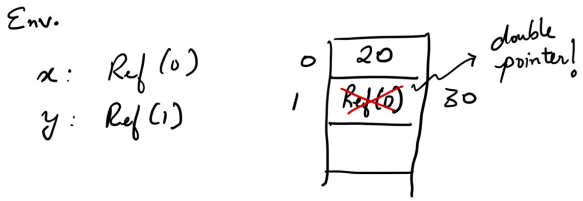
- Questions about these constructs?

What do these program do? Draw a picture to visualize execution of each.

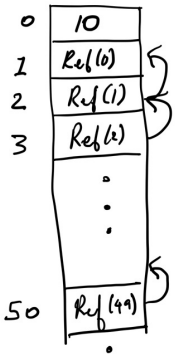
```
let x = newref(20) in
   let y = newref(30) in
     assignref(x, deref(y))
```




```
let x = newref(20) in
   let y = newref(x) in
     assignref(x, deref(y))
```




```
let rec crazy = function (n)
                  if (n <= 0)
                  then 10
                  else newref( f(n-1) )
   in
      crazy(50)
```



What about delete?

- C/C++ have new/delete operators.
- Why don't we have one in Lettuce?

- Memory management: programs ask for more memory as they execute and have memory that is no longer used.
  - If we did not delete/reclaim memory that is no longer needed, then we will inevitably run out.

Philosophy # 1: C/C++ Style

Philosophy # 2: Java/Scala/Python Style

What about Gangnam style??? ☺

Let’s delete Lettuce

- Let us add a delete functionality to memory in Lettuce.

- Over to notebook:
  - Syntax
  - Semantics
  - Problems??

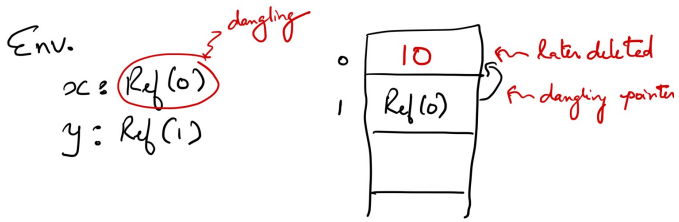
- Write some problematic programs?

Memory Self-Management issues

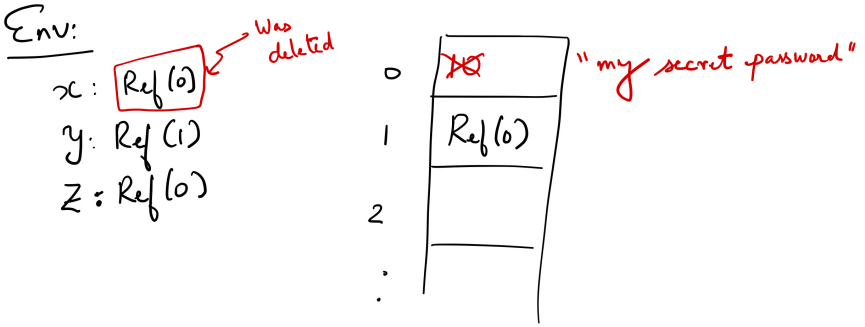
- Allow programmers to manage memory through new/delete operations.
- No checks when programmers access some memory through a pointer.
- many errors can happen.
  - These errors have led to serious security vulnerabilities including some you have heard of such as HEARTBLEED.

Dangling Pointers

```
let x = newref (10) in
  let y  = newref(x) in
    let _ = deleteref(x) in
      deref(deref(y))
```




```
let x = newref (10) in
  let y  = newref(x) in
    let _ = deleteref(x) in
      let z = newref("my secret password") in
        deref(deref(y))
```



Memory Leaks

```
let ref leakyLeeks = function (n)
                        if ( n <= 1)
                        then 1
                        else (
                              let var = newref (n) in
                              let rcall = leakyLeeks(n-1) in.
                                    deref(var) * rcall
                        ) in
 leakyLeeks (5)
```

Garbage Collection

- The programming language runtime takes care of managing memory for you.

- Advantages: no dangling pointers, no leaks, no mess, no fuss.
- Disadvantages: takes extra time, overhead on simple operations like assignments, function calls, etc.. Restricts programmer freedom.