CS 316 Final

Note Title 12/16/2009

1. a) F
   b) F, they are the same
   c) F, they should be the same if 1D. T if they're 2D
   d) F, compaction defrags the heap
   e) T
   f) F, Syntax describes grammatical structures
   g) T
   h) T, because it is a pointer
   i) T
   j) T
   k) T
   l) F, RTS controls function calls. ARs are simply function states
   m) F, Points to the end of the AR that called it.
   n) T
   o) F, Garbage collection is for the heap, not the AR
   p) T
   q) T
   r) T, Static variables are fixed in one location
   s) F, Heap and RTS are separate
   t) F, Everything is static
   u) T
   v) F Binding = Type Binding and can't undo in Statically-Typed
   w) T, Function Table is used dynamic inheritance
   x) T, Everything is a function call

2. a) 10-13-09 Notes - 2 branch conditional
   b)
   c)
   d)

3. a)
   b)

4.

5. a) Type checking @ compile time (statically)
   b) Faster to code, more flexible
   c) Java / C++
   d) PHP, Python, JavaScript

6. a) I, II, III (C++)
   b) I II III (Java)
   c) II (local vars in functions)
   d) II (formal params in functions)
   e) III
   f) I
   g) II
   h) II
   i) III - Because of explicit memory calls
   j) I
   k) III

I Static = Variables that don't change locations

II Stack-Dynamic = Depends on runtime

III Heap-Dynamic = Anything in the heap. Garbage collection

(7) (a) X
(b) Yes, `Node a = new Node();`
`Node m = new Node();`
`Node n;`
(c) Exercise Set #4, question 5 (and notes from 10-29-09)

(8) BaseAddress = address([5]) = 10
ElementSize = 2
Give the formula for address([i,])
[5, ..., 15] 5 = a_1, 15 = b_1
address([i,]) = BA + rank([i,]) × ES
address([i_1]) = 10 + (i_1 − 5) × 2

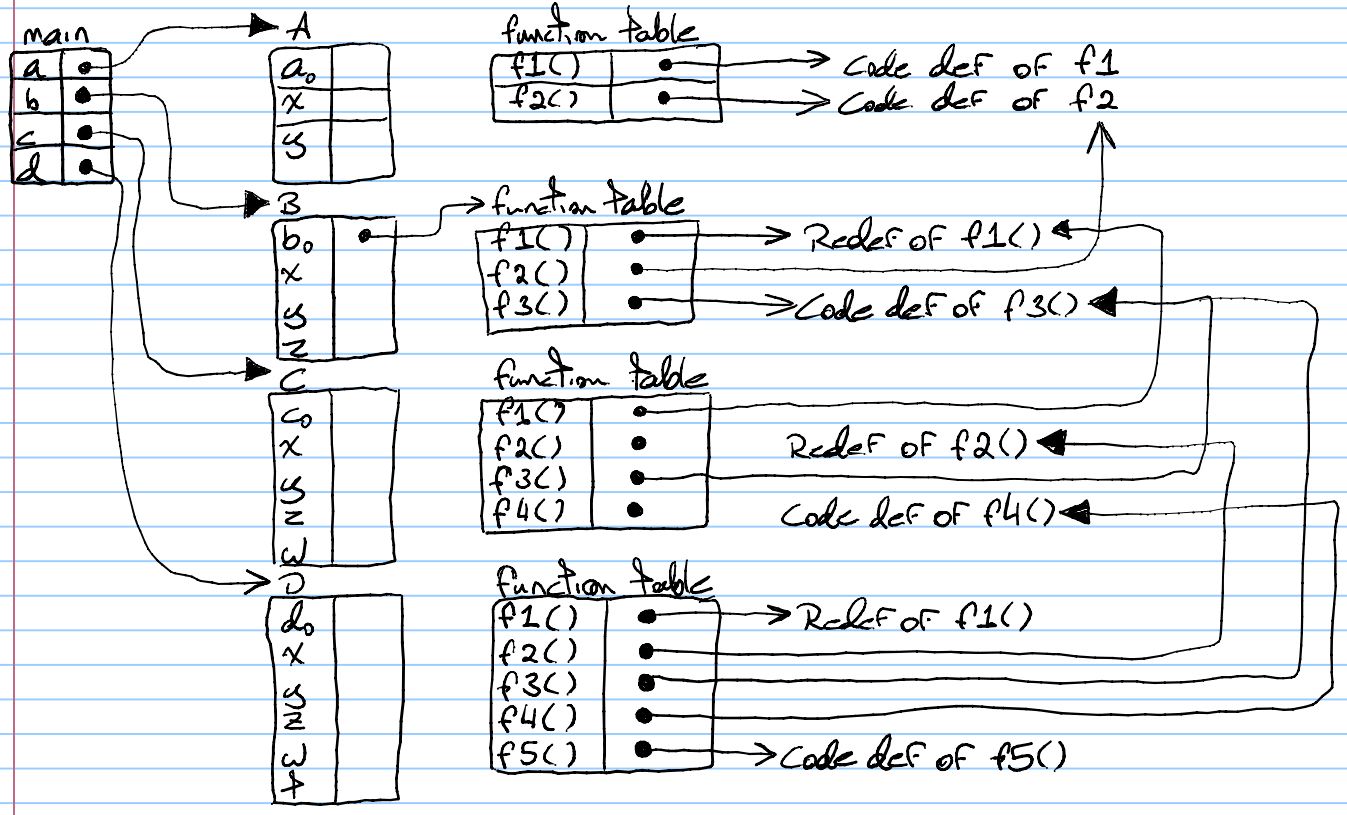
From: notes - 10
address([i_1, ..., i_n]) = BA + rank([i_1, ..., i_n]) × ES

For 1D array
rank([i_1]) = $i_1 - a_1$

(9) 2D array [0...10, 5...20]
BaseAddress = address([0,5]) = 10
ElementSize = 1
Compute address([5, 15])
Row Major = BA + ES * rowSize * ($i_1 - B_1$) + ES * ($i_2 - B_2$)
Column Major = BA + ES(($i_2 - B_2$) * colSize + ($i_1 - B_1$))

(10) base([]) = BA = the start address of the array [a_1... b_1]
base([i_1, ..., i_{k-1}]) = the value of the cell @ address([i_1, ..., i_{k-1}])
address([i_1, ..., i_{k-1}]) = base([i_1, ..., i_{k-1}]) + ($i_k - a_k$) × S, where S = ES if k = n, S = AS if k < n

(11) Notes from 11-12-09 (Person object, Name object, etc)



(12) From Notes 12

(a) Root nodes - static variables' memory cells, ARs within memory cells, registers, which contain references to heap cells

(b) The nodes are root nodes and non-root nodes.
The links are the references to the nodes

(c) Reachable = Nodes which are allocated and can be reached by a chain of 1 or more references from one of the root nodes.
Garbage = A node that isn't reachable. (If it loses a reference to the root)

(13) (a) (b)
(b) (2 ( )) or (2)
(c) (a)
(d) (b)
(e) (1 2 3)

- cons = constructs a list
- car = returns the first parameter of the input
- cdr = returns the second parameter of the input

Check Exercise Set #8 (or every parameter excluding the first)

(14) (a)
```
(define (append L1 L2)
        (if (null? L1) L2
            (cons (car L1) (append (cdr L1) L2))
        )
)
```
From 12-08-09

(b)
```
(define (map f list)
        (if (null? list) ( )
            (cons (f (car list)) (map f (cdr list)))
        )
)
```
From 12-08-09

(15)
```
length ( [ ], 0).
length ( H|T ) :- length (T, Y), L is Y+1.
```

(16)