Informatics 41 • Fall 2011 • David G. Kay • UC Irvine • Your name ________________________________

EIGHTH QUIZ

You have 15 minutes from the start of class to complete this quiz. Read the problems with care; work with deliberate speed. Don't give us more than we ask for. The usual instructions apply. Good luck!

Problem 1 (8 points)

We have distributed a Deus X reference sheet; you don't have to turn it back in, but we'll recycle it if you do. We'll have a better chance of assigning you partial credit if you show your work (e.g., draw a picture of the register(s) and/or memory locations).

(a) (2 points) Suppose that location 2222 of the Deus X machine's memory holds the number 100 and that location 3333 holds the number 75. What is in location 3333 after executing these three instructions? (The first number on each line indicates the instruction's address in memory.) Mark your final answer clearly.

```
0.   10  2222    (lda 2222)
1.    1  3333    (add 3333)
2.   20  3333    (sta 3333)
```

(b) (3 points) Suppose that location 5555 of the Deus X machine's memory holds the word `LINK`, that location 6666 holds the word `OINK`, and that location 7777 holds the word `PINK`. What does the deus X machine print after executing these instructions? Mark your final answer clearly.

```
0.   10  5555    (lda 5555)
1.   50  7777    (cmpa 7777)
2.   63     5    (jl 5)
3.    6  7777    (out 7777)
4.    7     6    (jmp 6)
5.    6  6666    (out 6666)
6.    8          (halt)
```

(c) (3 points) For each row **(a)** through **(h)** below, mark an X in the most appropriate column.

	Machine Language	Assembly Language	High-Level Language
(i) The Deus X examples we did in class			
(ii) Java or Python or Fortran or Scheme, for example			
(iii) Most tedious and error-prone			
(iv) Same instructions as machine language, but op codes and operands are more mnemonic (memory-aiding)			
(v) One statement may translate into many machine-language instructions			
(vi) Enables programmers to get the most code written in the least time			

Problem 2 (3 points)

For each of the algorithms or operations described below, check the box corresponding most closely to its complexity (i.e., its O-notation) in the average case.

(a) Inserting a new node into an empty binary search tree:

❑ Constant—O(1) ❑ Logarithmic—O(log n) ❑ Linear—O(n) ❑ Quadratic—O(n^2)

(b) Inserting a new restaurant into a (balanced) BST of n restaurants, ordered by name:

❑ Constant—O(1) ❑ Logarithmic—O(log n) ❑ Linear—O(n) ❑ Quadratic—O(n^2)

(c) Collecting an alphabetized list of restaurants from a BST of n restaurants, ordered by name:

❑ Constant—O(1) ❑ Logarithmic—O(log n) ❑ Linear—O(n) ❑ Quadratic—O(n^2)

Problem 3 (5 points)

Suppose we have a list called RL of restaurants defined as usual:

```
(define-struct rrant (name cuisine phone dish price))
```

For each of the following expressions, describe in one clear and precise English phrase what value it returns. Don't just say, "It does a foldr of plus and zero and ..."; give a description of what the expression *means*, something you could put in a software catalog so that a prospective buyer could find what he or she wanted. Use real-world terms, not program syntax terms: Say something like, "a list of the phone numbers of the restaurants that charge over $100," not "a list from mapping rrant-phone onto restaurants where the price field is greater than 100."

(a)
```
(map rrant-name (filter (lambda (R) (string=? (rrant-cuisine R) "Russian")) RL))
```

(b)
```
(local ((define L (map rrant-price
                       (filter (lambda (R) (member? (rrant-dish R)
                                                     (list "Turkey" "Duck" "Goose")))
                               RL))))
  (/ (foldr + 0 L) (length L)))
```

Problem 4 (4 points)

Using `map`, `filter`, and/or `foldr` as necessary, define the following function without using explicit recursion. [Hint: You may use `local` to define intermediate results or auxiliary/helper functions.]

```
;; moderate-rrants-by-cuisine:  list-of-rrant  string(cuisine)  ->  list-of-rrant
;; Return the rrants on the input list whose cuisine matches the input string and
;; whose price (of the best dish) is at least $10.00 and at most $25.00
(define moderate-rrants-by-cuisine
  (lambda (RL c)
```