Name: _____

1. (4 points) Give an example of a programming language for each category in this 2x2 matrix:

	Imperative	Functional
Static		
Dynamic		

2. (7 points) Draw a AST for the expression 2 + 3^2 * 4, assuming the usual operator precedence (here ^ is exponentiation).

3. (5 points) What would the following program print?

```
def f(x):
    return [ i**x for i in range(2, 6) ]

result = f(3)
print(result[2])
```

4. (8 points) Write Python code that has the same effect as this list comprehension, without using a list comprehension or map:

```
result = [ f(x) for x in range(n) ]
```

Name: _____

5. (8 points) Show the parse tree resulting from the input "`if x > y then x = 4`" using the following grammar:

$\langle stmt \rangle$ ::= $\langle ifstmt \rangle \mid \langle assignstmt \rangle$

$\langle ifstmt \rangle$ ::= if $\langle expr \rangle$ then $\langle stmt \rangle$

$\langle assignstmt \rangle$ ::= $\langle var \rangle = \langle expr \rangle$

$\langle expr \rangle$ ::= $\langle expr \rangle = \langle expr \rangle \mid \langle expr \rangle > \langle expr \rangle \mid \langle var \rangle \mid \langle num \rangle$

$\langle var \rangle$ ::= a | b | c | ... | y | z

$\langle num \rangle$ ::= 0 | 1 | 2 | ... | 9

6. (8 points) Recall the Church booleans in lambda calculus:

- $\lambda x.\lambda y.x \equiv$ true
- $\lambda x.\lambda y.y \equiv$ false

Write the Boolean "nand" function (which is true as long as at least one of its two inputs is false) in lambda calculus and show its operation on inputs "true false".

Name _____

7. (8 points) Draw the following list in terms of cons cells: (1 2 3) (4)

8. (8 points) Given the following Racket code:

```
(define f
  (lambda (x)
    (lambda (y) (+ (+ x 1) (* y 3)))))

(define g
  (lambda (x)
    (if (zero? x)
        (f 5)
        (lambda (y) (* x y)))))
```

What is the result of evaluating the following expressions? If the result is a function, just write "function" If there's an error, write "error"

(f 2) _____

((f 2) 4) _____

((g 2) 4) _____

(g 2) _____

((g 0) 4) _____

((g 4) 3) _____

9. (5 points) Write an expression that gives the third element of a list. Let using only car and cdr.

Name: _____

10. (8 points) Write a Racket function to count the number of 0's in a (non-nested) list using reduce (without using filter).

11. (8 points) Write a Racket function "append" that takes two lists and returns a single list with all elements of the first list (in order) followed by all elements of the second list (in order). That is, `(append '(1 2 (3)) '((4) (5 6))) = '(1 2 (3) (4) (5 6))`.

12. (10 points) Write a Racket function "curry" that converts a two argument function to a series of one argument functions, so that instead of calling it as `(f x y)`, we can call `(((curry f) x) y)`.

Name: _____

13. (10 points) Given the following Haskell code:

```
x :: Int
x = 42

y :: Int
y = 10

f :: Int -> Int -> Int -> Int
f x y z
  | x > y      = if x > z then x else z
  | x == y     = z
  | otherwise = f (x+2) z y

g :: Int -> Int
g 0 = 1
g 1 = 3
g n = n * g(n - 2)
```

What is the result of evaluating the following expressions? If the result is a function, just write "function". If there's an error or it doesn't produce a value, write "error".

`x > y` _____

`z / y` _____

`if x then 1 else 2` _____

`mod x y` _____

`f 3` _____

`f 5 4 3` _____

`map (f 5 4) [3..6]` _____

`f 3 4 5` _____

`g 3` _____

`g (-3)` _____

14. (5 points) Write a Haskell expression to count the number of 0's in a (non-nested) list using fold:

`count0s xs =`