CS1301 - Exam3 Name: ______________ Section or Grading TA: ________

Instructions:

- Please write clearly. What I cannot read, I will not grade.
- Show all your work in detail. I give partial credit.
- This exam has 10 pages including the title page. Please check to make sure all pages are included.
- This exam is closed book, closed notes, no calculators.
- Don't get bogged down on any one question. You will have 50 minutes to complete this exam.

I commit to uphold the ideals of honor and integrity by refusing to betray the trust bestowed upon me as a member of the Georgia Tech community.

Signature: ________________________________

Question	Points	Score
1. Vocabulary	15	
2. Recursion Fun	5	
3. Mystery Code	3	
4. Three Lambdas	6	
5. Lots-o-Letters	6	
6. Skip Every 3	8	
7. Factorial	6	
8. RedBlend	10	
9. BigO Graph	10	
Bonus Questions	0	
Total:	69	

Fall 2009

Vocabulary Questions

1. For each of the following vocabulary terms, write a concise 1-2 sentence definition. Be brief, and to the point.

   (a) (3 points) lambda

   Solution: A piece of code which can be executed as if it were a function but without a name. (Lambda is also a keyword used to create such an anonymous function.)

   (b) (3 points) global variable

   Solution: A global variable is a variable that can be seen (is visible) throught a program module. Defined outside of all functions.

   (c) (3 points) immutable data type

   Solution: A (compound) data type in which the elements can not be modified.

   (d) (3 points) runtime error

   Solution: An error raised by the python runtime while the program is executing if something goes wrong. For example, a divide by zero error.

   (e) (3 points) semantic error

   Solution: An error (in code) that leads to unexpected behavior. The program functions correctly (does what the code says) but the code does not actually perform the action that the programmer intended.

Code Understanding Questions

2. (5 points) Write down what would be `printed` when the following code is executed.

```
def rec(n):
    if(n<=1):
        return 1
    print n
    value=  rec(n-1) + rec(n-2)
    return value

result = rec(5)
```

Solution: 5
4
3
2
2
3
2

Grading:
5 points if they get all numbers correct and write them vertically. 4 points if they get all numbers correct, but don't write them vertically. 3 points if they miss only 1 number. 2 points if they get at least 4 of the numbers right (and/or add 0's and 1's) 1 point if they sort of see what is happening, but the numbers are all wrong.

3. (3 points) What does this mystery function do? Tell us the result of calling

```
mysteryFunc(5)


def mysteryFunc(n):    # n is an integer
   if n == 0:
      return 0
   else:
      return n + mysteryFunc(n-1)
```

> **Solution:** Answer: This function sums up the first N integers, or: "This function adds up all numbers from zero to N". Tthe number 15 proves they know how the function works.
>
> Grading:
> 3 points if they understand what it does, and give us 15. 2 points if they describe some of what it does, but give the wrong number. 1 point if they mention recursion, but don't know what it does. 0 points for anything else/less.

4. For each of the following lines, write what it returns:
   (a) (2 points) `map(lambda X: X / 2, range(6) )`

   > **Solution:** [0,0,1,1,2,2]

   (b) (2 points) `filter(lambda X: X < 3, range(6) )`

   > **Solution:** [0,1,2]

   (c) (2 points) `reduce(lambda X,Y: X + Y + 1, range(6) )`

   > **Solution:** 20

5. (6 points) Question: What is the output of the following code when executed?

```
print "A"
B = "C"
#print D

def rabbit(E):
    B = "F"
    for G in range(E):
        print "H"
        print "I"

    print B

    if E > 3:
        return "J"
        print "K"

    return "L"

def turtle(M):
    global B
    if M > 5:
        print "N"
    elif M > 4:
        print "O"

    if M > 3:
        print "P"
    else:
        print "Q"

    B = "R"
    return "S"

print "T"
print rabbit(2)
print B
print turtle(6)
print B
```

Solution: ATHIHIFLCNPSR Answer: ATHIHIFLCNPSR (printed vertically down!)
Grading Criteria: +1/2 point for each correct letter (up to 6). -1 point for EACH extra or missing letters. -1 points if the letters are not arranged vertically.

Code Writing Questions

6. (8 points) The following code takes in a .txt file with a name on each line. The function needs to separate these names into groups of 3 by inserting a newline after every 3rd name, and write this new output to another file. Please fill in the blanks to make this function accomplish that goal.

   Example Input and Output (Remember, your function must work for ANY input file NOT just this sample!)

Input File:	*Then the output would be:*
Sam	Sam
Peter	Peter
Micah	Micah
Melody	
Austin	Melody
Mallory	Austin
Jason	Mallory
Trevor	
	Jason
	Trevor

```
def separator(aFile):
    #Open the input file to read the names
    inFile = open(aFile, ____________)
    #The output file to write to
    outFile = open("output.txt", _____________)
    counter = 0
    #Read only the next line in the file
    name = inFile.__________________()

    while name != "":
        #Write the line to the output
        outFile.write(_____________)
        counter += 1
        #Here is where we check if we need to insert a separation
        if counter == ____________:
            #How do you separate the groups?
            outFile.write(_____________)
            #What happens to the counter?
            counter = _____________
        #read the next line
        name=inFile._________________()

    inFile.close()
    outFile.close()
```

Solution:

1. "r"
2. "w"
3. readline
4. name
5. 3
6."\n"
7. 0
8. readline

Grading: 1 point for each correct entry.

7. (6 points) Write a function named `factorial` that takes in an integer, and returns the factorial of that number. The factorial of a number ($n!$) is defined as being the product of all positive integers less than or equal to n. Mathematically that is $n! = \prod_{x=1}^{n} x$. Another way to write this is:

$$n! = \begin{cases} 1 & \text{if } n = 0; \\ n \times (n-1)! & \text{if } n > 1; \end{cases}$$

So the factorial of 5 is $5 \times 4 \times 3 \times 2 \times 1 = 120$. Note that the factorial of zero is defined to be 1. You may assume that the number you will be given will be non-negative. You may solve this using a loop, recursion or any other method.

Solution:

```
def factorial( aNum):
   if (aNum == 0):
     return 1
  return aNum * factorial(aNum-1)
```

or

```
def factorial( aNum):
   product = 1
   index = 1
   while(index <= aNum):
      product = product * index;

   return( product )
```

Grading: 1 point for a correct header. 2 points for returning one if aNum is zero 2 points for calculating the correct answer. 1 point for returning the correct answer.

8. (10 points) Write a function named `redBlend` that accepts two pictures as parameters. It should set the red values in the first picture to the red value of the corresponding pixel in the second picture. You may assume that both pictures are the same size. Because you are modifying the first picture parameter, you do not need to return anything.

   Some functions that might be helpful for this question:

   - `getPixels(picture)`
   - `getPixel(picture, x, y)`
   - `getWidth(picture)`
   - `getHeight(picture)`
   - `getX(pixel)`
   - `getY(pixel)`
   - `getRed(pixel)`
   - `setRed(pixel,value)`

Solution:

```
def redBlend(p1, p2):
   for px1 in getPixels(p1):
      px2 = getPixel( p2, getX(px1), getY(px2)  )
      setRed(px1, getRed(px2))
```

Grading: 2 points for correct header, 2 points for looping through all pixels in the first image. 4 points for finding the corresponding pixel in the 2nd image. 1 points for getting the red value from picture two, 1 point for setting the red value in picture one to match. (-1 if they return something other than None) -1 If they accidentally modify picture 1 instead of picture 2 but are otherwise correct)

9. (10 points) Draw a single graph with a line for each of the four main Big O complexity classes we have learned about. Label both the X and Y axes appropriately. For each line, label it with the Big O complexity class it represents, and also give the name of an algorithm that falls in that complexity class.

Solution:

Grading: 1 point for each line that is correctly labeled with a Big O complexity class 1 point for each algorithm that matches the big O complexity class (not necessarily the line) 1 point for the labels on the axes

O(n) - Sequential or Linear search O(n^2) - Bubble Sort or Insertion sort O($nlogn$) - Merge Sort (or Quicksort) O($logn$) - Binary search

X-axis: - Number of items, elements, etc Y-axis: - Work, time, or number of comparisons

10. (3 points (bonus)) In this semester we have had 3 individual assignments (Face, Avoid Walls, My Program) and 6 pair assignments (Tips & Conversions, Base Conversions, Dance & Menu, Yellow Wall, Special FX, and Robot Movie)

    (a) Which (if any) pair assignments do you wish were made into individual assignments?

    (b) Which (if any) individual assignents would you prefer be made into a pair assignment?

    (c) Do you think that the added overhead in partner assignents (of scheduling meeting times, and meeting with partners) was worthwhile? Did having a partner to work with allow you to finish the assignments faster, or learn more, or have more confidence in your understanding? Why or why not?