Name : _____

Grading TA: _____

- INTEGRITY: By taking this exam, you pledge that this is your work and you have neither given nor received inappropriate help during the taking of this exam in compliance with the Academic Honor Code of Georgia Tech. Do NOT sign nor take this exam if you do not agree with the honor code.
- DEVICES: If your cell phone, pager, PDA, beeper, iPod, or similar item goes off during the exam, you will lose 10 points on this exam. Turn all such devices off and put them away now. You cannot have them on your desk.
- ACADEMIC MISCONDUCT: Academic misconduct will not be tolerated. You are to uphold the honor and integrity bestowed upon you by the Georgia Institute of Technology.
  - Keep your eyes on your own paper.
  - Do your best to prevent anyone else from seeing your work.
  - Do NOT communicate with anyone other than a proctor for ANY reason in ANY language in ANY manner.
  - Do NOT share ANYTHING during the exam. (This includes no sharing of pencils, paper, erasers).
  - Follow directions given by the proctor(s).
  - Stop all writing when told to stop. Failure to stop writing on this exam when told to do so is academic misconduct.
  - Do not use notes, books, calculators, etc during the exam.
- TIME: Don't get bogged down by any one question. If you get stuck, move on to the next problem and come back once you have completed all of the other problems. This exam has 12 questions on 18 pages including the title page. Please check to make sure all pages are included. You will have 1 hour and 45 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. I have also read and understand the requirements outlined above.

Signature: _____

Question	Points	Score
1. Vocabulary	12	
2. Fill in the Blank	6	
3. Multiple Choice	5	
4. Short Answer	13	
5. Short Answer	6	
6. Filter and Map	5	
7. Code Fixing	8	
8. Ascii Art	9	
9. Sorting	9	
10. isSorted	5	
11. BigO Graph	10	
12. Guess Number	12	
Total:	100	

1. *(12 points)*
   For the following vocabulary term, write a concise 1-2 sentence definition. Be brief, and to the point.

   (a) [3 pts] 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.

   (b) [3 pts] syntax error

   Solution: An error in a program that makes it impossible to parse – and therefore impossible to interpret.

   (c) [3 pts] 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.

   (d) [3 pts] block

Solution: One or more program statements that share the same level of indentation.

2. *(6 points)*
Complete each statement below by filling in the blank:

1. A _____ loop iterates through all items in a sequence.
2. A _____ is a named entity that can refer to data or functions.
3. You can select a _____ out of a list by using a colon inside brackets, such as aList[3:5]
4. You use _____ in your python programs, denoted by the # symbol, to explain in natural language how your program works.
5. In python, the single equal sign is used for _____, while the double equal sign is used for _____.

Solution: Grading: +1 for each correct answer.
For
Variable or Identifier
Slice
Comments
Assignment, Equality Checking

3. *(5 points)*
For each of the following questions, select the appropriate answer by circling it.

(a) [1 pt] `x = input("Please enter a number.")` What is `type(x)`?
A. None B. bool C. float D. int **E. str** F. list

(b) [1 pt] Which of these commands detects something behind the robot (non-fluke side)?
A. getObstacle() **B. getIR()** C. getBright() D. getLight() E. getStall()

(c) [1 pt] Which of these statements evaluates to True?
A. True or False and False
B. True and False or False
C. True and True and False
D. not True or False

(d) [1 pt] What happens when you try to keep reading lines after you've hit the end of the file?
A. You get an EndOfFile error
B. You get an empty string.
C. The file loops back to the top and starts reading again.

D. You start getting junk data from the places in memory that come after the file.

E. None of the above

(e) [1 pt] What happens if you try to take a slice that extends past the end of a string? (e.g. `"hello"[0:50]`)

A. IndexError

B. A slice up to and including the last character in the string.

C. The slice of the string plus junk data from the next places in memory.

D. The string loops back around and gives you multiple copies of the string.

4. *(13 points)*

For each of the following questions, give a brief answer:

(a) [1 pt] What is printed when the following lines of code are evaluated?

```
s = "CS 1301 rocks!"
print( s[ :7:2] )
```

Solution: Answer: C 31

(b) [5 pts] The open function has ____(number) modes that you can use to open a file, and they are: _____.
The default mode for the open function when no mode is specified is: _____.

Solution: Answers: 3
read or r
write or w
append or a
Default: read or r

Grading: +1 for each correct entry

(c) [3 pts] What is printed when the following lines of code are evaluated? Be sure to format your output exactly as Python would.

```
l = ["open", "close", "in", "out", "up", "down" ]
for i in range(0,6,2):
    print( l[i])
```

Solution: Answer:
open
in
up

Grading: +1 for each word. (-1 if they don't write them vertically, or for each incorrect word.)

(d) [4 pts] Assume each of the following lines is entered in the shell one at a time. Circle any of the lines that produces an error. If no errors are found then write what is printed (from a and b).

```
a = [[[1, 2, 3, 4, 5],[6,7,8]],9]
a[-1] = 200
b = a[:]
b[0][0][3] = 17

print( a )
```

```
print( b )
```

Solution:
[[[1, 2, 3, 17, 5], [6, 7, 8]], 200]
[[[1, 2, 3, 17, 5], [6, 7, 8]], 200]
Grading: +2 for each line...

5. *(6 points)*

For each of the following questions, give a brief answer:

(a) [4 pts] A text file named "fruits.txt" contains the following three lines:

Apple
Blueberry
Cantaloupe

The following code is run:

```
f = open("fruits.txt", "r")
f.readline()
s = f.readline()
f.close()

f = open("output.txt", "w")
f.write(s + "\n")
f.write("Orange\n")
f.close()
```

What does the file "output.txt" contain?

Solution: Answer:
Blueberry
Orange

Grading: +2 for each correct line. -1 for each incorrect line, -1 if they don't draw them vertically, or indicate the newline character.

(b) [2 pts] What is a boolean expression?

Solution: "An expression that evalutes to a boolean value"

Grading: +2 if correct. +1 if they say something about true/false but don't mention expressions or boolean conditionals (`<`, `<=`, `>`, `>=`, `==`, `!=`), or if they mention expressions or boolean conditionals but don't mention true/false.

6. *(5 points)*

```
myList = [1,2,3,4,5]
while (len(myList) > 0):
    print myList
    myList = filter(lambda x: (x-1) > 0, myList)
    myList = map(lambda y: y-1, myList)
```

Examine the code above. Write down exactly what is printed when the above code is executed:

Solution:

```
[1, 2, 3, 4, 5]
[1, 2, 3, 4]
[1, 2, 3]
[1, 2]
[1]
```

Grading:
+5 if exactly correct.
+4 if almost exactly correct but off by one number.
+3 if the lists get smaller each time.
+1 if they write at least one list.

7. *(8 points)*

The following code was written by Andrew to read a file, searching through it for all integers, and then return the largest integer. The input file may be any file containing any text (not necessarily numbers). After writing the code below, Andrew had to run it 4 times. Each time he ran it, he got an exception (see below) and he debugged a single problem each time. After fixing the four errors, the code worked correctly. For each of the 4 traceback errors we provide below, fix a single line of the original code so that Andrew's function will work (HINT: the problematic line is usually clearly indicated in the error). Note that each error can be fixed by changing a single line, but other solutions may be accepted if the program maintains its original functionality.

```
1 |def findNums(infile):
2 |    # load the file, read content
3 |    inputFile = open(infile, 'r')
4 |    readText = read(inputFile)
5 |    # filter-out only the numbers
6 |    filteredText = ""
7 |    for char in range(readText):
8 |        if char not in "0987654321":
9 |            filteredText += " "
10|        else:
11|           filteredText += char
12|    listOfNumStrings = filteredText.split()
13|    # convert to ints
14|    listOfInts = map(lambda num: int(num), someList)
15|    # return max
16|    return listOfInts.max()
17|
18|# extra code that runs the program
19|print findNums('test.txt')
```

(a) [2 pts]
```
Traceback (most recent call last):
  File "C:/Python27/findNums.py", line 4, in findNums
    readText = read(inputFile)
NameError: global name 'read' is not defined
```

Solution: `inputFile.read()` - + 2pts

(b) [2 pts]
```
Traceback (most recent call last):
  File "C:/Python27/findNums.py", line 7, in findNums
    for char in range(readText):
TypeError: range() integer end argument expected, got str.
```

Solution: `for char in readText` - +2 pts

(c) [2 pts]
```
Traceback (most recent call last):
  File "C:/Python27/findNums.py", line 14, in findNums
    listOfInts = map(lambda num: int(num), someList)
NameError: global name 'someList' is not defined
```

Solution: listOfInts = map(lambda num: int(num), listOfNumStrings)

(d) [2 pts]
```
Traceback (most recent call last):
  File "C:/Python27/findNums.py", line 16, in findNums
    return listOfInts.max()
AttributeError: 'list' object has no attribute 'max'
```

Solution: return max(listOfInts)

8. *(9 points)*

The asciiArt function, defined below, takes in two parameters: A myro picture object, and the name of a file to write (as a string). The function will loop through all of the pixels of the picture and calculate their brightness. Brightness is defined as the average of the red, green and blue values. If a pixel is "bright" (average is above 128) the function should write a space character (" ") to the file. For every "dark" pixel (average at or below 128) the function should write a hash character ("#") to the file. The function will also need to write some newline characters to the file so that every row in the picture corresponds to a line in the file.

You need to fill in the blanks to make the asciiArt function work as described above:

```
def asciiArt(aPic, outputFileName):

   outFile = ___________________________

   for y in _____________________:

      for x in _____________________:

         pixel = ______________________________
         r,g,b = getRGB(pixel)

         brightness = ___________________________
         if brightness <= 128:

            ______________________

         else:

            ______________________

      ____________________

   ________________
```

Solution:

```
def asciiArt(aPic, outputFileName):
   outFile = open(outputFileName, "w")
   for y in range(getHeight(aPic)):
      for x in range(getWidth(aPic)):
         pixel = getPixel(aPic, x, y)
         r,g,b = getRGB(pixel)
         brightness = (r + g + b) / 3
         if brightness <= 128:
            outFile.write("#")
         else:
            outFile.write(" ")
       outFile.write("\n")
   outFile.close()
```

Grading: +1 for each correct blank.

9. *(9 points)*
Here is a sequence of numbers: 4,7,1,2,9,0,3

(a) [3 pts] Illustrate how a bubble-sort would sort the above list of numbers. You do not need to show each swap, simply show the numbers after each pass. underline the numbers that are guaranteed to be in sorted order. Do all passes, even if the list becomes sorted before the algorithm would finish.

Solution: 4, 1, 2, 7, 0, 3, 9
1,2,4,0,3, 7,9
1,3,0,3, 4, 7 , 9
1, 0, 2, 3, 4, 7, 9
etc...

Grading:
+3 points for sorting the list with a recognizable bubblesort algorithm.
Misc things to take off for:
- Don't bubble the front half of the list correctly -1
- Skipped the last 3 passes (which do not change the list) -1
- Didn't underline correct numbers: -1

(b) [3 pts] Illustrate how an insertion-sort would sort the above list of numbers. After each pass, underline the numbers that are guaranteed to be in sorted order.

Solution: 4,7,1,2,9,0,3
1,4,7 2,9,0,3
1,2,4,7 9,0,3
1,2,4,7,9 0, 3
and so on...
Grading:
3 points for doing insertion sort correctly.
2 points for doing selection sort instead of insertion sort.
0 points for not doing it right.

(c) [3 pts] Illustrate how a merge-sort would sort the above list of numbers.

Solution: Step 1 : break them up into 7 lists of one element each.
Step 2 : merge into pairs (leaving one out)
step 3 : merge 2's into 4's (single can remain alone, or you can merge the single into a 3 if you want.)
Step 4 : merge the rest together and underline.

Grading:
3 points for doing it right. -1 point for starting with lists of size 2 instead of size 1 -1 point if they "forget" a number to get an even number of items.

10. *(5 points)*
Write a function named `isSorted` that takes in a list of numbers. (Numbers are either integers or floats.) If the list is sorted your function should return True. If any number in the list is not in sorted order, your function must return False. Your function should also return True for lists of length 0 or 1.

Example test cases:

```
>>> result =  isSorted( [1,2,3,4])
>>> result
True
>>> isSorted( [4.8, 2.3])
False
```

Solution:

```
def isSorted(aList):
   for i in range(len(aList)-1):
      if aList[i] > aList[i+1]:
         return False
   return True
```

Grading: 1 point for a correct header.
1 point for looking at each number.
1 point checks if numbers on the left are smaller. 1 point returns False if the list isn't sorted.
1 point for returning True if the list is sorted.

11. *(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

12. *(12 points)*
Write a function called **guessNumber** that accepts no parameters. Your function should prompt the user to enter a single digit between 0 and 9, using input("Enter a single digit between 0 and 9"). When the user enters a single digit as a string, look it up in the dictionary (d). If the digit the user entered is in the dictionary, print the value associated with the key and return. If the digit the user entered is NOT in the dictionary, print a message "Sorry, that is not a correct digit!" and then call guessNumber() again recursively.

You may assume the following dictionary is a global variable:

```
d = { "0" : "Very Small!", "2": "Nice and Even",
           "7" : "Lucky!", "9": "Very Big!"}
```

Solution:

```
def guessNumber():
   user_input = input("Enter a single digit between 0 and 9:")
   if user_input in d :
      print( d[userInput] )
   else:
      print( "Sorry, that is not a correct digit!")
      guessNumber()
```

Grading: 2 points for a correct def header.
2 points for getting the user input
2 points for checking to see if the dictionary has a matching key.
2 points for printing the correct response (if it exists).
2 points for printing the "Sorry" messsage (if it doesn't exist).
2 points for the recursive call (if it doesn't exist.)

This page intentionally left blank. You may use it as scratch paper. If you place an answer on this page, box it, label it clearly, and indicate clearly on the original problem page that your answer is on this page.