Name : ______________________________
Section 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 6 questions on 6 pages including the title page. Please check to make sure all pages are included. 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. I have also read and understand the requirements outlined above.

Signature: ______________________________

Question	Points	Score
1. Vocabulary	6	
2. Multiple Choice	8	
3. BigO Blanks	4	
4. Divide By 2	5	
5. Every Other	10	
6. grayScale	10	
Total:	43	

1. *(6 points)*
For each of the following vocabulary terms, write a concise 1-2 sentence definition. Be brief, and to the point.

   (a) [3 pts] argument

   Solution: argument - A value provided to a function when the function is called. This value is assigned to the corresponding parameter in the function.

   (b) [3 pts] parameter

   Solution: parameter - A name used inside a function to refer to the value passed as an argument.

2. *(8 points)*
For each of the following multiple choice questions, indicate the most correct answer! Indicate your selected answer by circling it.

   (a) [1 pt] Convert 110101_2 to decimal (base 10):
   A. 29 B. 52 **C. 53** D. 54 E. 118

   (b) [1 pt] Convert 129_{10} to binary (base 2):
   A. 01000011 B. 10000011 C. 01000111 D. 11001101 **E. 10000001**
   F. 01010101

   (c) [1 pt] Which of the following python expressions is invalid? Assume that all named variables exist, refer to data of the type implied by their name, and all indexed subelements exist.

   A. `aList[2] = aList[4]+aList[5]`
   B. `dictionary1.append(23)`
   C. `tupleA = tupleA+(3,)`
   D. `aNum = 24+ord('a')`
   E. All statements above are valid.

(d) [1 pt] Assume you have an empty dictionary created as follows:

```
aDict = {}
```

Which one of the following statements would **not** raise an exception?

A. `aDict[ ["1"] ]= "one"`

B. `aDict.items()`

C. `aDict[aDict]="two"`

D. `aDict={1;1}`

(e) [1 pt] Examine the following code:

```
def foo(aList):
  for i in range( len(aList) // 2  - 1 ):
    for j in range(len(aList) - 2):
       aList[i-j] = 0
```

Let N be the length of aList. If your "unit of work" is assigning a zero to a list element `aList[i-j]`, consider how the number of units of work increase as N increases. What is the Big O complexity class of the function foo?

A. $O(N)$ **B.** $O(N^2)$ C. $O(2^N)$ D. $O(LogN)$ E. $O(NlogN)$

(f) [1 pt] Given the following code, what is printed when it executes?

```
p = "foo"
temp = ""
temp2 = ""
for x in range(len(p)):
  for y in range(len(p)):
    temp = temp + p[x]
    temp2 = temp2+p[y]
print(temp2)
```

A. `fffooo`

B. `fffoooooo`

C. `foofoofoo`

D. `oofoofoof`

E. None of the above

(g) [1 pt] Opening a file in write mode creates the file if it doesn't exist.

A. True B. False

(h) [1 pt] What is stored in aTuple after this code is run:

```
aTuple = (1,2,3,['hello','world'], True)
aTuple[3][1] = False
```

A. Error: You can't index into an int

B. `(1,2,3,['hello',False],True)`

C. `(1,2,3,[False,'world'],True)`

D. `(1,2,3,['hello','world'],False)`

E. Error: You can't change that element in a tuple

3. *(4 points)*
Give an example of a searching or sorting algorithm that falls within each Big O time complexity class:

1. $O(LogN)$ ____________________
2. $O(N)$ ____________________
3. $O(NLogN)$ ____________________
4. $O(N^2)$ ____________________

Solution: Grading: +1 for each correct answer.
Binary Search (of a sorted list)
Linear Search
Merge/Quick sort
Bubble/insertion sort

4. *(5 points)*
Entirely re-write the following code using functional programing instead of iteration. Your code must replace every number in `aList` with the result of dividing that number by two. You may **not** use a for loop or while loop (traditional iteration) or recursion. You may use lambda expressions or define a helper function.

```
for n in range( len(aList)):
   aList[n] = aList[n] / 2
```

Solution:

```
aList = map( lambda x: x/2, aList)
```

Grading: +2 for the correct lambda or helper function +2 for the correct call to map +1 for re-assigning the returned list to aList

5. *(10 points)*
Write a function, **everyOther**, which will take in two parameters: fromFile and toFile. fromFile is a string containing the file name on your computer you wish to read from. toFile is a string of the file name on your computer you wish to write to. You should read in the contents of fromFile, and write every other line to toFile. For example, if fromFile contains:

```
Hi
Isn't
CS
Fun
```

Then after your function runs, toFile should contain:

```
Hi
CS
```

Solution:

```
Example solution:

def everyOther(fromFile, toFile):
    fromF = open(fromFile)
    toF = open(toFile, "w")
    myLines = fromF.readlines()
    for i in range(0, len(myLines)):
        if i % 2 == 0:
            toF.write(myLines[i])
    fromF.close()
    toF.close()


Grading Rubric:
+1 opens fromFile correctly
+1 opens toFile with "w" as mode
+2 correctly reads lines from file
+2 correctly writes lines to file
+2 writes only every other line to file
+2 closes both files correctly
```

6. *(10 points)*
Write a function named `grayScale` that accepts a picture as a parameter. Your function should convert the picture to a gray scale image. (all pixels are some shade of gray).

To do this, you should take the red, green, and blue values for each pixel, find their average, and place that value back into all three color channels (r,g,b) of the pixel.

Because it modifies the picture that was passed in, your function should not return anything (None). You may assume that all of the appropriate functions have already been imported from Myro.

Solution:

```
def grayScale(p):
   for pix in getPixels(p):
      r = getRed(pix)
      g = getGreen(pix)
      b = getBlue(pix)
      a = r+b+g // 3
      setRed(pix,a)
      setGreen(pix,a)
      setBlue(pix,a)

Grading:
  +1 for correct header
  +2 for iterating through all pixels.
  +2 for getting the r,g,b colors correctly.
  +2 for calculating the average correctly (ok, but bad form to do non-int division
  +2 for putting the average back into all three r,g,b channels.
  +1 if they correctly return None (deliberately, or by default)
```