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 7 questions on 8 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	9	
2. Code Understanding	4	
3. Range and Modulo	5	
4. OnlyAs	10	
5. noTs	8	
6. Seek Light	8	
7. Create Usernames	12	
Total:	56	

1. *(9 points)*
   For each of the following vocabulary terms, write a concise 1-2 sentence definition. Be brief, and to the point.

   (a) [3 pts] compound data type

   Solution: A data type that can hold primitive data types, such as a list, tuple, or dictionary.

   (b) [3 pts] global variable

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

   (c) [3 pts] sequence

   Solution: Any of the data types that consist of an ordered set of elements, such as strings, lists or tuples.

2. *(4 points)*
   Assume the following code was entered into the IDLE window. Explain the difference between b and c.

```
a = [1,2,3,4]
b = a
c = a[:]
```

Solution: B is an alias of A, while C is a copy or clone of A. If you modify an item using b[x] = newItem A will also change because it is pointing to the same data. If you modify something using C[x] = newItem, A will not change.

Grading:
+1 point for using term alias +1 point for using term clone or copy +2 points for explaing how an alias is different from a copy.

3. *(5 points)*
   What does the following code PRINT when executed?

```
for n in range(1,10):
  if (n%3 == 0) or (n%5 == 0):
     print "oak"
  elif (n%4 == 0):
     print "elm"
  if (n==5):
     print "birch"
```

Solution:

```
oak
elm
oak
birch
oak
elm
oak
```

Grading: +5 for all correct. Otherwise, -1 for each missing or extra item. (zero minimum)

4. *(10 points)*
   The function below accepts a file name (as a string) and writes every line in that file that begins with a capital letter A to a new file named newFile.txt.

   Fill in the blanks appropriately:

```
def onlyAs(filename):
   inF  = _______________  ( ___________ , ______ )
   outF = ______________   ( ___________ , ______ )

   lines = inF.________________________
   for line in lines:
      if _________________________________________________________:
         outF.________________________________

   inF.close()
   outF.close()
```

Solution:

```
def onlyAs(filename):
   inF = open( filename, "r")
   outF = open( "newFile.txt", "w")

   lines = inF.readlines()
   for line in lines:
      if line[0] == "A":  #or line.startswith("A")
         outF.write(line)

   inF.close()
   outF.close()
```

Grading: 1 point for each correctly filled in blank, with the exception of the if conditional. They get +2 points if that one is correct, +1 if it is close to correct.

5. *(8 points)*
Write a function called `noTs` that accepts a list (of strings) as a parameter. It should return a new list that only contans the strings that do not contain the letter T (either an uppercase T or a lowercase t).

For example:

```
>>> result = noTs( ['Georgia Tech', 'Buzz', 'Yellow', 'Jackets',
   "Sting 'em!", "Ramblin' Wreck" ] )
>>> print result
 ['Buzz', 'Yellow', "Ramblin' Wreck"]
```

Solution:

```
def noTs( aList ):
  retList = []
  for string in aList:
    # also works: if "t" not in string and "T" not in string:
    if not ("t" in string or "T" in string):
       retList.append(string)
  return retList
```

Grading:
1pt - Correct def statement
2pts - traverses the items in the list
2pts - checks each item for T and for t
2pts - adds non-t items to the new list.
1pt - returns the new list.

6. *(8 points)*
Write a function named `seekLight` that accepts an integer as a parameter. The integer corresponds to how long your function should run, in seconds, before returning.

While running, your function should get the value of the robots' center light sensor (using a `getLight('center')` call). If the value returned is smaller than 500, your robot should move forward at full speed for one second. If the light sensor value is larger than (or equal to) 500, the robot should beep at 800Hz for 1 second, and not move.

Make sure that, no matter what, your robot stops and the function returns when the time is up! You may assume that the myro library is already loaded and that your robot is already connected (init has been called) to the computer.

When your function returns, it should return None.

Solution:

```
def seekLight(time):
  while timeRemaining(time):
     if getLight("center") >= 500:
        beep(1,800)
     else:
        forward(1,1)
```

Grading: 1 point for correct header
1 point for running N seconds
1 point for checking the getLight sensor each itteration
2 points for moving forward when triggered
2 points for beeping when not triggered
(-2 points (instead of -4) if they reverse the beep and move)
1 point for function returning (None) when the time is up.
Misc Penalties:
-1 if doesn't stop at end of function.
-1 mixing-up time and frequency parameters beep(800,1)

7. *(12 points)*
Write a function called `createUsernames` that accepts three lists as parameters. The first parameter will contain strings which contain first names. The second parameter will be a list of strings that contain last names. The third list will contain integers. The function should combine each item in the lists to create a username. A username is made up of the first letter of the first name, up to the first six letters of the last name, and the digit. Note that if a last name has less than six letters the entire last name will be used.

Your funciton should return a list of usernames in the same order as the input data. You may assume that all three lists are of the same length.

For example:

```
>>> results = createUsernames(["Glenn","Ikenna"], ["HollingsWorth","Omekam"],
     [7,3] )
>>> print results
['GHollin7', 'IOmekam3']
```

Solution:

```
def createUusername(aList, aList2, aList3):
    newList = []
    for index in range( len(aList) ):
        string = aList[index][0] + aList2[index][:6] + str(aList3[index])
        newList.append(string)
    return newList
```

Grading:
1pt - Correct def statement
2pts - uses only first initial of first name.
2pts - Uses up to 6 characters of last name.
(-1 if they only grab 5 characters instead of six...)
2pts - Converts digit to a string.
2pts - concatenates each item the resulting username string.
2pts - Adds each username to a list.
1pt - returns the list of usernames.