UCSB CS8 CS8 F18 Mirza
Gauchospace Gradescope Piazza

Name: [redacted] (as it would appear on official course roster)	
Umail address: [redacted]	@umail.ucsb.edu

1

e03
CS8 F18

EXAM: e03: Final Exam

ready?	date	points
true	Tue 12/11 08:00AM	100

You may not collaborate on this exam with anyone. If you need to use the restroom, you must leave your cell phone with the exam proctor before leaving the room.

- Write your name at the top of this page **AND EVERY ODD NUMBERED PAGE.**
- Double check that you turned in ALL pages; look for "End of Exam" on the last page.
- This exam is **closed book, closed notes, closed mouth, cell phone off.**
- You are permitted **one sheet of paper** (max size 8.5x11") on which to write notes.
- This sheet will be collected with the exam, and might not be returned.
- Please write your name on your notes sheet.

NOTE: All references to **Python** on this exam mean **Python 3**, so you should answer accordingly.

1. Consider the definitions of the functions foo and bar given below:

1, 2

```
def foo(x):
    ll = []
    for i in range(1,3):
        ll.append(x*i)
    return ll
```

1, 2   1, 5

```
def bar(x):
    ll = []
    for i in range(1,3):
        ll.append(x*i)
        if type(x) == int or type(x) == float:
            x = x - 1
    return ll
```

3

In the table provided, write the value returned by each of the following calls to foo and bar. Write Error if the call results in an error or exception. Write None if nothing is returned by the function.

pts	function call	write result here	pts	function call	write result here
(2 pts)	foo(4)	[4, 8]	(2 pts)	bar(4)	[4, 6]
(2 pts)	foo(2.5)	[2.5, 5]	(2 pts)	bar(2.5)	[2.5, 3]
(2 pts)	foo("foo")	['foo', 'foofoo']	(2 pts)	bar("foo")	['foo', 'foofoo']
(2 pts)	foo(-1)	[-1, -2]	(2 pts)	bar(-1)	[-1, -4]

2. Consider the following definition for the function fun:

```
def fun(x):
    if type(x) != list:  ✓
       return False
    count = 0
    for item in x:
      if type(item)!= str:
         return True
      if len(item) > 2:
         count = count + 1
    return count
```

In the table provided, write the value returned by each of the following calls to fun. Write "Error" if the call results in an error or exception. Write "None" if nothing is returned by the function.

pts	function call	write result here
(2 pts)	fun(['300', '20'])	1
(2 pts)	fun(["-1", (1, 2), 3])	True
(2 pts)	fun(["ox", "cow", "camel"])	2
(2 pts)	fun("ox") False	False
(2 pts)	fun(["ox", "cow", "parrot", "dog"])	3
(2 pts)	fun([])	0

3. Consider the following Python functions:

```
def foo_1a(aList):
    result = ''
    for x in aList:
        if (type(x)==str and 'o' in x):
            result = result + x
    return result
```

```
def foo_1b(aList):
    result = ''
    for x in aList:
        if (type(x)==str and 'o' in x):
            result = result + x
    print(result) # Print vs return
```

```
def foo_2a(aList):
    result = ''
    for x in aList:
        if (type(x)==str and 'o' in x):
            result = result + x
        return result # Pay attention to indentation
```

```
def foo_2b(aList):
    result = ''
    for x in aList:
        if (type(x)==str and 'o' in x):
            result = result + x
        print(result) # Print vs return
```

```
def foo_3a(aList):
    result = ''
    for x in aList:
        if (type(x)==str and 'o' in x):
            result = result + x
            return result # Pay attention to indentation
```

```
def foo_3b(aList):
    result = ''
    for x in aList:
        if (type(x)==str and 'o' in x):
            result = result + x
            print(result) # Print vs return
```

In the table provided below write the output of calling each of the functions from the Python shell. Write "Error" if the call results in an error or exception. Write "N/A" if nothing is returned or printed by the function.

pts	function call	write result here	pts	function call	write result here
(2 pts)	`foo_1a(["ox", 5,"cow"])`	'oxcow'	(2 pts)	`foo_1b(["ox", 5,"cow"])`	'oxcow'
(2 pts)	`foo_2a(["ox", 5,"cow"])`	'ox'	(2 pts)	`foo_2b(["ox", 5,"cow"])`	'ox' 'ox' 'oxcow'
(2 pts)	`foo_3a(["ox", 5,"cow"])`	'ox'	(2 pts)	`foo_3b(["ox", 5,"cow"])`	'ox' 'oxcow'

2

e03
CS8 F18

4. (2 pts) In lab07 (Scrabble word finder), a Python dictionary was used store all the letters of the alphabet and their corresponding point values. What was the rationale behind this? *Circle the single best answer below:*

   a. It made it easy to identify valid words (i.e. words that exist in the English dictionary)
   [✓ b.] It made it easy to calculate the score for a given word
   c. It made it easy to read all the valid words from a file
   d. All of the above
   e. None of the above

5. The code below is intended to read data from a file which contains words, exactly one word per line. Write the function `readFile` which take two inputs: the file name `fname` and `N`. Your program should store the words from line number N to the end of the file in a list and return the list. If the file contains less than N lines your function should return an empty list. You may assume that the file exists.

For example consider a file called `words.txt` that contains the following words:

```
cow   0
cat   1
dog
duck
owl
mouse
```

Suppose that we call the function `readFile` as follows: `l1 = readFile('words.txt', 4)` The list `l1` should contain all the words starting at line number 4 until the end of the file: `['duck', 'owl', 'mouse']`.

The code below was tested, and then several lines of code were removed.

3

(20 pts) Fill in the blanks in a way that is consistent with the comments.

```
# function: readFile
# Inputs:
# fname: name of the file to read from
# N: Line number in the file
# read from a file containing words, one per line.
# return a list of all the words starting at line number N in the file

def readFile(fname, N):

    # (a) Open the file
    infile = open(fname, 'r')

    # Initialize an empty list
    wordList = []

    # (b) Read the first N-1 lines from the file
    for i in range(N-1):
        infile.readline()

    # (c) Read the remainder of the file and append each line to wordList
    for line in infile:
        wordList.append(line.strip())

    # close file
    infile.close()

    # (d) return the answer
    return wordList
```

6. (8 pts) Implement a function called `coinToss` that tosses a coin N times and returns the percentage of time the outcome was a 'head'. The function should take the value of N as parameter and return a number between 0 and 100. Assume N is always positive.

Hint: This function is similar to the function `rollDie` that you wrote for lab06. Instead of rolling dies, you are now tossing a coin. Tossing a coin results in one of two outcomes: head or tail. Map these outcomes to specific numbers in your random experiment. Use the Python's `random` module.

```
import random

def coinToss(N):
    ''' Return the percentage of heads observed when you simulate tossing a coin N times'''
    # Write your code here
    numhead = 0
    for i in range(N)
        if random.randrange(0,2) == 1:
            numhead += 1
    return numhead / N * 100
```

7. (10 pts) Write a function to find the shortest word in a list of words. If more than one word has the same length as the shortest word return the word that appears earlier in the list. Assume that the parameter passed to the function is always a non-empty list containing only strings. You may choose to implement the function recursively or iteratively (using loops).

```
def shortestWord(myList):
    '''
        Example usage (1): shortestWord(['table', 'chair','axe', 'wood'])
        Returns: 'axe'
        Example usage (2): shortestWord(['table', 'car','axe', 'wood'])
        Returns: 'car'
    '''
    shortestWord = myList[0]
    for i in myList:
        if len(i) < len(shortestWord):
            shortestWord = i
    return shortestWord
```

Name: [redacted] (as it would appear on official course roster)	
Umail address:	[redacted] @umail.ucsb.edu

3

e03

CS8 F18

8. (10 pts) Write a recursive function called `countOddWords` that takes a list of words as input and returns the number of words in the list that have an odd valued length. If an element in the list is not a string, don't include it in your count. If the parameter passed to the function is not a list return 0.

```
def countOddWords(alist):
    '''
        returns the number of words whose length is odd
        Example usage (1): countOddWords(['table', 'chair','axe', 'wood'])
        returns: 3
        Example usage (2): countOddWords(['tractor', 5 ,'axe', 'wood'])
        returns: 2
    '''
    if type(alist) != list or alist == [ ]:
        return 0
    if type('alist[0]) != str:
        return countOddWords (alist[1:])

    if (len(alist[0]) %2) == 1:
        return 1+ countOddWords(alist[1:])
    else:
        return countOddWords(alist [1:])

                    ↓
```

~~cleaned up version:~~

```
def countOddWords(alist):
    if type(alist) != list or alist == [ ]:
        return 0
    if (type(alist[0]) == str) and (len(alist[0] %2)==1)
        return 1 + countOddWords(alist[1:])
    return countOddWords(alist[1:])
```

9. (10 pts) Write a function called `getEndangeredBirds` that takes as input a Python dictionary called `birds`. The parameter `birds` contains bird names and boolean values indicating whether a bird is endangered or not. The function should return a list of bird names that are endangered.

For example:

```
getEndangeredBirds({'falcon': True, 'owl': True, 'eagle': False, 'cowbird': True})
```

returns `['falcon', 'owl', 'cowbird']`

```
def getEndangeredBirds(birds):
  '''return a list of bird names that are endangered
  i.e. return a list of keys in the dictionary birds
  whose value is True'''
   endangered = [ ]
   listbirds = birds.items()
   for tuple in listbirds:
       if   tuple[1] == True:
            endangered.append( tuple [0])
   return  endgngered
```

End of Exam