CS 1301 Exam 2 Version A
Spring Semester 2018

Name (print clearly including your first and last name):

____

GT account username (gtg, gth, msmith3, etc):

____

Signature:

____

- 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.
- Signing and/or taking this exam signifies you are aware of and in accordance with the **Academic Honor Code of Georgia Tech** and the **Georgia Tech Code of Conduct**.
- 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).
    - <u>Stop all writing when told to stop. Failure to stop writing on this exam when told to do so is academic misconduct.</u>
    - Do not use notes, books, calculators, etc during the exam.
    - You are not allowed to leave the exam with your exam paper for any reason.
- Devices: If ANY electronic device 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.
- Extra paper is not allowed. If you have exhausted all space on this test, talk with your instructor.
- Pens/pencils and erasers are allowed. Do not share.
- All code must be in Python.

1. [20 pts] Pretend you are the python interpreter. Evaluate each of the expressions below. Write down the value that they evaluate to, and the type of that value in the provided columns. If the expression is not in valid python syntax, or will throw an exception, simply write "Error". If your answer is a string include quotations around your answer (i.e "hello"). If your answer is a float, make sure you include the decimal (i.e 5.0). Hint: Think of assigning the expression to a variable, and ask yourself these questions: What is the value of that variable? (Column 1) What is the type of that variable? (Column 2)

Table 1 Expression Table

Expression	**Value of Result [1 pt]**	**Type of Result [1 pt]**
len({1:5,1:9,1:15})		
(3) + (1)		
[“Boo”, “Burrito”][1][:2]		
len((1, 2) + (3, 4))		
{4 : “DootDoot”, 8 : “Bubble”, 1 : “Cake”}[1] + “Pop”		
([1, 4], [1, 2, 3])[1][2]		
“I am hungry”.split()		
[1, 2, 3][3]		
{1 : “Uno”, 2 : “Deux”, 3 : “San”}[1:]		
{“Panda” : 5, “Sloth” : 2}[“Sloth”] + 2		

2. [21 pts] For each of the following multiple choice questions, indicate the best answer by **completely** filling in the circle located at the left of the letter.

a) [3 pts] How would you change the dictionary referenced by the variable a_dict from **{1: [1, 2, 3], 2: (1, 2, 3)}** to **{1: [1, 2, 3, 4], 2: (1, 2)}**?

O A.

```
a_dict[1] += 4
a_dict[2] -= 3,
```

O B.

```
a_dict[1].append([4])
a_dict[2] -= 3
```

O C.

```
a_dict[1].append(4)
a_dict[2] = a_dict[2] - 3,
```

O D.

```
a_dict[1].append(4)
a_dict[2] = a_dict[2][:-1]
```

O E.

```
a_dict[1].append([4])
a_dict[2] = a_dict[2][:1]
```

b) [3 pts] What is the value of my_list after the following lines of code are executed?

```
a_list = ['a', 'b', [2, 4]]
b_list = a_list[:]
my_list = b_list
a_list[2].append(4)
a_list[1] = 'c'
b_list[0] = 'z'
```

O A. ['z', 'c', [2, 4, 4]]

O B. ['a', 'b', [2, 4, 4]]

O C. ['z', 'b', [2, 4, 4]]

O D. ['z', 'b', [2, 4]]

O E. ['a', 'c', [2, 4]]

c) [3 pts] Which of the of the following statements about catching exceptions is **FALSE**?

○ A. You should put code inside of a try block if it will potentially cause a runtime error.

○ B. You should put code that will handle exceptions inside of an except block.

○ C. You cannot have try / except blocks inside of loops.

○ D. If an exception is thrown in an except block, the code in the finally block will get executed before the program crashes.

○ E. You can have nested try / except blocks.

d) [3 pts] How many lines will the file 'file.txt' contain after the following lines of code are executed?

```
f = open('file.txt', 'w')
f.write('dog')
f.write('cat')
f.close()
f2 = open('file.txt')
x = f2.readlines()
f2.close()
f3 = open('file.txt', 'w')
f3.write(x[0])
f3.write('\nthe\nend\n')
f3.close()
```

Answer choices:

○ A. 1

○ B. 3

○ C. 5

○ D. 7

○ E. None of the above.

e) [3 pts] Which of the following is **NOT** a legal expression in python?

- ○ A. `my_dict = {1: {1: 2, 3: 4, (3, 4): 4}}`
- ○ B. `my_dict = {'1': [1], [2]: '2', '3':[3]}`
- ○ C. `my_dict = {{1: 2}[1]: 3, (1, 2)[0]: 4}`
- ○ D. `my_dict = {print('hi'): print('bye')}`
- ○ E. All of the above are legal expressions.

f) [3 pts] Which of the following statements describes scope correctly in regards to the code below?

```
def func(a):
    t = 0
    for x in a:
        t += x
    return t
```

- ○ A. t is outside of a for loop, so it is a global variable.
- ○ B. x is a global variable because it is used inside of a loop.
- ○ C. a is a parameter, so it is a global variable.
- ○ D. a is a parameter, so it is a local variable.
- ○ E. None of the above statements are correct.

g) [3 pts] Assume you have imported the math library using the statement **from math import *** How could you print out the value of pi?

- ○ A. `print(math.pi())`
- ○ B. `print(math.pi)`
- ○ C. `print(pi)`
- ○ D. `print(pi())`
- ○ E. None of the above

4. [15 pts] Tracing: Indicate **in the box provided** what will be printed to the Python shell when the following blocks of code are executed. Only the response written **inside the box** will be graded.

a) [5 pts]

```
alist = [0,0]
blist = [1,1]
def list_magic(alist):
    print(alist)
    for num in range(len(alist)):
        alist.append([2])

list_magic(alist)
list_magic(blist)
print(alist)
print(blist)
```

b) [5 pts]

```
a = []
b = ['CS']
b.append(a)
c = b[:]
a += [3]
c[1].append('A')
b[1] = 'B'
print(a)
print(b)
print(c)
```

c) [5 pts]

```
a_dict = {print('a'): 'b'}
print(a_dict[print('c')])
b_dict = {'d' : a_dict}
c_list = []
b_dict['d'][c_list.append(4)] = 'f'
print(a_dict)
print(c_list)
```

5. [12 pts]
Write a function called squadUp that takes in two parameters: a list of tuples and a power level (represented by an integer). The tuples will always be composed of one name (a string) and a power level (an integer). Your goal is to create a list of names that satisfy your power level requirement. Only add names associated with power levels **higher** (exclusive) than the power level given to your list. Return the list of everyone whose power level satisfies this constraint. See examples below for further clarification.

Note: If no one satisfies the requirements, return an empty list.

```
Example #1:
    >>> squadUp([("AgarWhale", 6), ("SriCar", 0), ("SheTall", 2)], 0)
Expected Output #1:
    ['AgarWhale', 'SheTall']

Example #2:
    >>> squadUp([("AbsoluteMax", 8), ("N", 2), ("Json", 5), ("Alex", 10)], 4)
Expected Output #2:
    ['AbsoluteMax', 'Json', 'Alex']
```

6. [14 pts]
Write a function called bestFriend that takes in one parameter: a dictionary with keys representing names as strings mapped to values as tuples. The tuples will contain potential best friends’ names as strings. The last name in the tuple will represent the name of the best friend (for the respective key). Return a dictionary that maps each person to his or her best friend from the tuple. See examples below for clarification.

Note: All tuples will contain at least one name.

Example #1:
```
    >>> bestFriend({"Irene" : ("Wendy", "Joy", "Yeri"), "Carl" : ("Russell", 
"Dug", "Kevin", "Ellie"), "Grac" :("Cat",)})
```
Expected Output #1:
```
    {'Irene': 'Yeri', 'Carl': 'Ellie', 'Grac': 'Cat'}
```

Example #2:
```
    >>> bestFriend({"Baymax" : ("Hiro", "Tadashi"), "Black Panther" : ("Shuri", 
"W'Kabi")})
```
Expected Output #2:
```
    {'Baymax': 'Tadashi', 'Black Panther': "W'Kabi"}
```

7. [18 pts]
Write a function called yelpElite that takes in a csv filename as a parameter. This file is organized into three columns and any number of rows as well as one header row, as shown below. The first column will hold names of yelp members. The second column will hold the years that the member has been elite for. The third column will hold the number of thumbs up that the member has received. Write the names of members who have been elite for **more than** five years and who have received **more than** 100 thumbs up to a new text file named "bestCritics.txt". Each name should appear on a new line.

file.csv

Name	Years Elite	Thumbs Up
Kalen Allen	1	900
Gordon Ramsay	51	10001
Gael Greene	5	6800
Guy Fieri	7	10000
Master Eater	100	100

Example:

```
>>> yelpElite("file.csv")
```

Expected Output in bestCritics.txt:

```
Gordon Ramsay
Guy Fieri
```