

# CS 1301 Exam 2

## Fall Semester 2021

### Version A

Name (print clearly including your first and last name as it appears on your Buzzcard):

---

Signature indicating you understand the Georgia Tech Honor Policy:

---

GTID# (903000000, etc.):

---

- 
- ❖ 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.
    - Follow directions given by the proctor(s).
    - Stop all writing when told to stop. Continuing to write after the end of the exam is an honor violation.
    - Do not use notes, books, calculators, phones, laptops etc. during the exam.
    - You are not allowed to leave the exam with your exam paper for any reason.
  - ❖ Devices: If your cell phone, smartwatch, laptop, 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.
  - ❖ 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) **TABLE COMPLETION** [24 pts] (3 points each) Pretend you are the python interpreter. Evaluate each of the expressions below. Write down the value that each evaluates to. If your answer is a string, include quotes around your answer (i.e "hello"). If your answer is a floating point number make sure you include the decimal (i.e 5.0). Write the word error in both columns if the expression causes an error.

**Table 1: Expression**

Expression	Return Value of Expression (2 pts)	Data Type of Expression (1 pt)
("P",) + ("S",) + ("L",)		
len(["pumpkin",["pecan","apple"]])		
"treat" in {"trick": "treat"}		
{"eat", "more"}:"candy"		
(("halloween",), 800)[0][0]		
{"ghost": "Casper", "Maine": [4, 5, 1]}["Maine"][2]		
["costumes"].append("ghost")		
{print("scary"): "hrs", 2: ";p"}[None]*2		

2. **MULTIPLE CHOICE** [33 pts] (3 pts each) For each multiple choice question below, indicate the best answer by filling in the corresponding circle.

a) How many lines are printed to the shell after the following code is run?

```
spookyActs = [("pumpkin patch", 2), ("haunted house", 5),  
("pumpkin carving", 0), ("graveyard walk", 7)]  
for activity, count in spookyActs:  
    if count % 2 == 0:  
        continue  
    else:  
        print(count)
```

- ☐ A. 4
- ☐ B. 3
- ☐ C. 2
- ☐ D. 1
- ☐ E. 0

b) Which of the following expressions would change the contents of **myList** from ["candy corn", "m&ms", "kitkat"] to ["candy corn", "m&ms", ["reeses"]]?

- ☐ A. myList.append("reeses")
- ☐ B. myList.remove("kitkat")
- ☐ C. myList = myList[0:2].append(["reeses"])
- ☐ D. myList[2] = ["reeses"]
- ☐ E. None of the above

c) Which of the following is a valid way of printing pi from the math module given the following import statement?

```
import math as x
```

- ☐ A. `print(math.pi)`
- ☐ B. `print(x.pi())`
- ☐ C. `print(pi)`
- ☐ D. `print(x.pi)`
- ☐ E. Both C and D

d) Given `aDict = {"Costume": {"Fairy" : ("TinkerBell", 43.22), "Superhero": ["Hulk", 74.53]}}`, what is `aDict["Costume"]["Superhero"][1] * 2`?

- ☐ A. "HulkHulk"
- ☐ B. ["Hulk", 74.53, "Hulk", 74.53]
- ☐ C. 149.06
- ☐ D. "TinkerBellTinkerBell"
- ☐ E. aDict is not a valid dictionary.

e) Lists are \_\_\_\_\_. Tuples are \_\_\_\_\_. Dictionary keys are \_\_\_\_\_.

- ☐ A. Immutable, Mutable, Mutable
- ☐ B. Mutable, Mutable, Mutable
- ☐ C. Mutable, Immutable, Immutable
- ☐ D. Mutable, Immutable, Mutable
- ☐ E. None of the above

f) What is the value of `aDict` after the following code is run?

```
aList = ["boo", "scary", "mask", "mask", "scary"]
aDict = {}
for index, word in enumerate(aList):
    aDict[word] = aList[index]
```

- ☐ A. {0: 'boo', 1: 'scary', 2: 'mask', 3: 'mask', 4: 'scary'}
- ☐ B. {'boo': 'boo', 'scary': 'scary', 'mask': 'mask'}
- ☐ C. {0: 'boo', 1: 'scary', 2: 'mask'}
- ☐ D. {0: 'scary', 1: 'boo', 2: 'mask', 3: 'mask', 4: 'scary'}
- ☐ E. None of the above

g) How many lines are printed to the shell after the following code is run?

```
halloweenSnacks = ["pretzels", "punch", "chips"]
i = 5
for word in halloweenSnacks:
    try:
        print(word[i])
    except:
        i -= 1
    finally:
        print(word)
```

- ☐ A. 2
- ☐ B. 3
- ☐ C. 4
- ☐ D. 5
- ☐ E. 6

h) Which of the following could be a possible value that the random module function **randrange()** returns given the following implementation:

```
random.randrange(8, 32, 4)
```

- ☐ A. 32
- ☐ B. 13
- ☐ C. 18
- ☐ D. 20
- ☐ E. None of the above.

Use the following code to answer parts i, j, and k.

```
aList = ["pumpkin", "patch", "Friday the 13th", 2020]
cList = aList
bList = aList[:]
aList[2] = aList[2] + 1
cList[0][1] = "full moon"
bList.append("scooby")
cList[0] = "nightmare"
```

i) What is the value of **aList** after the code above is run?

- ☐ A. ["nightmare", "Friday the 13th", 2021]
- ☐ B. ["pumpkin", "full moon", "Friday the 13th", 2021, "scooby"]
- ☐ C. ["pumpkin", "full moon", "Friday the 13th", 2020]
- ☐ D. ["nightmare", "Friday the 13th", 2021, "scooby"]
- ☐ E. None (NoneType)

j) What is the value of **bList** after the code above is run?

- ☐ A. ["nightmare", "Friday the 13th", 2020]
- ☐ B. ["pumpkin", "full moon", "Friday the 13th", 2020, "scooby"]
- ☐ C. ["pumpkin", "full moon", "Friday the 13th", 2021]
- ☐ D. ["nightmare", "Friday the 13th", 2021, "scooby"]
- ☐ E. ["pumpkin", "patch", "Friday the 13th", 2020, "scooby"]

k) What is the value of `cList` after the code above is run?


- ☐ A. `["nightmare", "Friday the 13th", 2021]`
- ☐ B. `[["pumpkin", "full moon"], "Friday the 13th", 2021, "scooby"]`
- ☐ C. `[["pumpkin", "full moon"], "Friday the 13th", 2021]`
- ☐ D. `["nightmare", "Friday the 13th", 2021, "scooby"]`
- ☐ E. `None (NoneType)`

3) **Tracing** [12 pts] (4 points each) Show exactly what would be printed out when each of the following segments of code are executed. None of these code segments will cause an error. They all have at least partial output that would be shown.

a) 

```
def fallFun(events):
    eventList = []
    for e in events.keys():
        if e % 5 == 0:
            print(events[e] + " is fun!")
            eventList.append(e)
        elif events[e] == 17:
            print("Too scary!")
    return eventList

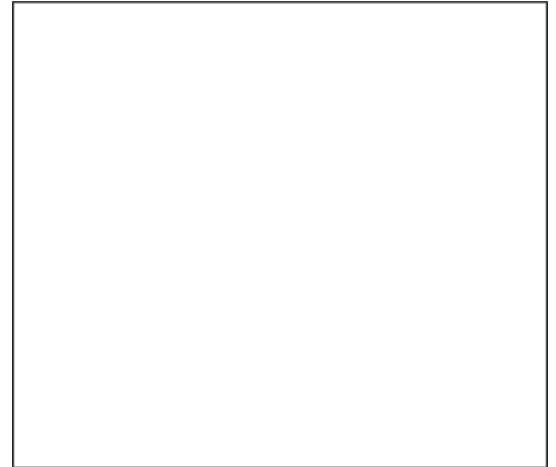
events = {15: "Costume Contest", 17: "Corn
Maze", 40: "Apple Bobbing"}
print(fallFun(events))
```





```
b) def stumes(aList):
    for i, j in aList:
        try:
            new = str(i) + j
            print(new)
        except:
            print("dope stume bro")
        finally:
            return j

    print(stumes([("Mario Batali", 9),
("Astronaut", 4), ("Epic Sax Guy", 8)]))
```



```
c) def traceMe(aTup):
    spookyToop = aTup + ("october vibez",)
    (a, b, c) = spookyToop
    c = b[:]
    a[0] = c
    newTup = (b, c, a)
    print(newTup)

    traceMe(("Reese's", "Not sorry",
"ghostface"))
```



CODING [31 pts]

**CODING 1** [8 pts] - Write a function called `topStumes()` that takes in one parameter: a list of tuples. Each tuple includes a costume name (str) and a rating out of 10 (int). The function should return a dictionary mapping costumes to their ratings if the costume has a rating that is greater than or equal to 5.

Example Output #1:

```
>>> stumes = [("Mario Batali", 3), ("Astronaut", 8), ("Epic Sax Guy", 2),
("Gollum", 5)]
>>> topStumes(stumes)
{'Astronaut': 8, 'Gollum': 5}
```

Example Output #2:

```
>>> stumes = [("Mario Batali", 9), ("Michael Myers", 2), ("Chucky", 5),
("Dracula", 10)]
>>> topStumes(stumes)
{'Mario Batali': 9, 'Chucky': 5, 'Dracula': 10}
```

**CODING 2** [10 pts] - Write a function called **trickOrTreat()** that takes in a list of strings containing trick-or-treating locations in the form: "building room". The function should return a dictionary of buildings mapped to a list of the room numbers in that building that you can buy candy from (ints).

Example Output #1:

```
>>> locations = ["Skiles 154", "CULC 101", "Skiles 233", "Kendeda 102"]
>>> trickOrTreat(locations)
{'Skiles': [154, 233], 'CULC': [101], 'Kendeda': [102]}
```

Example Output #2:

```
>>> locations = ["Klaus 145", "MRDC 115", "IC 201", "IC 117"]
>>> trickOrTreat(locations)
{'Klaus': [145], 'MRDC': [115], 'IC': [201, 117]}
```

**CODING 3** [13 pts] - Write a function called **candyHaul()** that takes in one parameter: a dictionary of people. Each key is a person's name (str) mapped to a list of tuples containing a candy name (str) and candy quantity (int). The function should return the name of the person who received the most candy.

**Note:** No two people will receive the same quantity of candy.

Example Output #1:

```
>>> candyDict = {"Parul": [("skittles", 5), ("starburst", 7)], "Jakob" :  
[("snickers", 2), ("reeses", 8)], "Nelson": [("butterfinger", 12), ("sour  
patch", 3)]}  
>>> candyHaul(candyDict)  
"Nelson"
```

Example Output #2:

```
>>> candyDict = {"Parul": [("skittles", 5), ("starburst", 7)], "Jakob" :  
[("snickers", 5), ("reeses", 1)]}  
>>> candyHaul(candyDict)  
"Parul"
```