

15-110 Midterm #2a – Fall 2018
50 minutes

Name: _____

Andrew ID: _____@andrew.cmu.edu

Section: _____

- You may not use any books, notes, or electronic devices during this exam.
- You may not ask questions about the exam except for language clarifications.
- Show your work on the exam (not scratch paper) to receive credit.
- If you use scratch paper, you must submit it with your andrew id on it, and we will ignore it.
- All code samples run without crashing. Assume any imports are already included as required.
- Do not use these post-midterm2 topics: recursion, etc.

DO NOT WRITE IN THIS AREA		
Part 1 (Very Short Answers)	25 points	
Part 2 (CT)	30 points	
Part 3 (FR / checkKalado)	15 points	
Part 4 (FR / maxDataValue)	10 points	
Part 5 (FR / coin flips)	20 points	
Part 6/bonus	5 points bonus	
Total	100 points	

1. [25 pts; 2.5 pts each] Very Short Answers

Answer each of the following very briefly.

- 1) About how many passes does mergesort require to sort a list of N numbers?
- 2) About how many total steps (not just passes) does selectionsort require to sort a list of N numbers?
- 3) What general topic did David Danks discuss in his guest lecture?
- 4) If M is a list of length 10, this line works just fine:
 `M[3] = 'x'`
But if M is a string of length 10, that same line will crash. Why?
- 5) What general topic did Jesse Schell discuss in his guest lecture?

6) In all our Monte Carlo examples, what do we have to do to get a more accurate result?

7) If a function $f(L)$ takes a 1d list L and always returns `None`, which seems more likely: that f is destructive or non-destructive?

8) What general topic did Lorrie Cranor discuss in her guest lecture?

9) What can you do faster with a set of N unique numbers than with a list of the same N numbers?

10) What can you do with a list of N unique numbers that you cannot do with a set of the same N numbers?

2. [30 pts; 10 pts each] Code Tracing

Indicate what each will print. Place your answer in the boxes below each block of code. Show your work, outside the box, for partial credit.

```
def ct1(L):
    M = [ ]
    n = len(L)
    for k in range(n):
        M.append(L[k][n-1-k])
        if (k%2 == 0):
            M[-1] *= -1
    return M

print(ct1([[1, 2, 3], [4, 5, 6], [7, 8, 9]]))
```

```
def ct2(d):
    s = set()
    t = set()
    for key in d:
        value = d[key]
        if (value in s):
            t.add(value)
        s.add(value)
    return sorted(t)

print(ct2({1:5, 2:3, 5:3, 4:5, 7:5, 6:4}))
```

```
def ct3(n):
    x = 0
    y = 0
    for i in range(n):
        try:
            d = i%10
            if (1/d > 0):
                x += 1
        except:
            y += 1
    return [x, y]
print(ct3(25))
```

3. [15 pts] Free Response: **checkKalado(L)**

Here, we will invent the game Kalado, which is a simplified variant of Kaladont. In Kalado, each word has to have at least one letter, and each word (except the first word) has to start with the last character of the previous word, without regard to case (so 'A' and 'a' match). With this in mind, write the function `checkKalado(L)` that takes a list of words `L`, and returns `True` if it is a legal Kalado list, and `False` otherwise.

Notes:

- * If the list does not have at least 2 words, then it's not a legal Kalado list.
- * If any word has no characters, then it's not a legal Kalado list.

Here are some sample test cases:

```
assert(checkKalado(['ab', 'bc', 'cda', 'ab']) == True)
assert(checkKalado(['ab', 'bc', 'ab']) == False)
assert(checkKalado(['AB', 'bc']) == True)
assert(checkKalado(['AB', '']) == False)
assert(checkKalado(['AB']) == False)
```

4. **[10 pts] Free Response: `maxDataValue(path)`**

For this exercise, you may assume the CSV files will contain one header row, followed by rows that only contain comma-separated integers. You may also assume every CSV file contains at least one row and one column of data values. With that in mind, write the function `maxDataValue(path)` that takes a path to a CSV file, and returns the largest value that occurs in the data.

Hint: you will want to call `readCsvFile(path)` (which you do not have to write!). Remember: `readCsvFile(path)` takes a path and returns the data in that csv file as a 2d list.

5. [20 pts] Free Response: coin flips

Write these two functions:

oddsOfAllSameFlips(n, trials)

This function takes two integers, *n* and *trials*, both of which you may assume to be positive, and uses Monte Carlo techniques to compute and return the odds that if you flip a coin *n* times you'll either get *n* heads or *n* tails. So `oddsOfAllSameFlips(3, 100)` would run 100 trials to find the odds that 3 coin flips would result in either 3 heads or 3 tails.

trialSucceeds(n)

This is a helper function you need to write that runs a single trial using *n* coin flips. You should call this helper function in `oddsOfAllSameFlips` (the other function you are writing here).

Notes:

- * You may assume `flipCoin()` is written for you, and randomly returns 'H' or 'T'
- * You must use Monte Carlo techniques to receive any credit here, even if you know how to compute the answer directly some other way.

[You may use the next page for this question]

[This page intentionally blank]

[Exam continues on next page]

6. Bonus/Optional: [5 pts, 2.5 pts each] What will this print? Clearly circle your answer.

```
def bonusCt1(n):
    s = ''
    for i in range(1,n):
        try: s += str(float(1/i))[i]
        except: s += '1'
    return n+float(s)
print(bonusCt1(5))
```

```
def bonusCt2(L):
    while (len(L)<10):
        for i in range(L[-2]):
            L.append(sum(L)%100)
    return L[-L[0]]
print(bonusCt2([1,2,3]))
```