15-112
Summer 2018 Final Exam
June 22, 2018

Name:

Andrew ID:

Recitation 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 to receive credit.
- You may complete the problems in any order you'd like; you may wish to start with the last four problems, which are worth most of the credit.
- All code samples run without crashing. Assume any imports are already included as required.

Don't write anything in the table below.

Question	Points	Score
1	13	
2	7	
3	6	
4	12	
5	12	
6	15	
7	15	
8	20	
Total:	100	

1. **Code Tracing**

   Indicate what each will print. Place your answer (and nothing else) in the box below each block of code.

   (a) (8 points) CT1

```
import copy
def ct1(L):
    a = L[0]
    b = copy.copy(L[0])
    c = copy.deepcopy(L[0])
    d = L
    d[1] = 112
    a[1] = 113
    c[1] = 114
    b[0][0] = 115
    a[0][1] = 116
    c[0][0] = 117
    print("a: ", a)
    print("b: ", b)
    print("c: ", c)
    print("d: ", d)
    print("L1: ", L)

L = [[[2,3], 8], 4]
ct1(L)
print("L2: ", L)
```

(b) (5 points) CT2

```
import string
def ct2(s):
    result = ""
    for i in range(0, len(s) - 2):
        if s[i] in string.digits:
            n = int(s[i])
            result += n * s[i+1]
        if s[i] == s[i+1] == s[i+2]:
            result += s[i].lower()
        elif s[i] == s[i+1]:
            result += s[i].upper()
    return result

print(ct2("aaabc2zy2gb"))
```

2. (7 points) **Reasoning Over Code**

Find an argument (the value of L) for roc1 that makes it return True. Place your answer (and nothing else) in the box below the code.

```
def roc1(L):
    z = [0,0]
    n = roc1Helper(L,z)
    return n == 12 and z[0] == 3 and z[1] == 6

def roc1Helper(L,z):
    if L == []:
        z[0] += 1
        return 0
    if type(L[0]) == list:
        return roc1Helper(L[0],z) + roc1Helper(L[1:], z)
    if type(L[0]) == int:
        z[1] += 1
        return L[0] + roc1Helper(L[1:], z)
```

3. **Short Answer**

   Answer each of the following *very briefly*.

   (a) (2 points) Name 2 ways that sets and lists differ. Note: Answers that appeal to syntax and method naming will not be accepted.

   (b) (2 points) List 2 style rules from the 112 style guide, and then write a code example that violates the 2 rules.

   (c) (2 points) Write 1-2 lines of code that would crash precisely because strings are immutable.

4. (12 points) **Big-O:** What is the Big-O runtime of each of the following in terms of n?

Built-in Big-O Runtimes					
Strings		Lists		Sets	
`len(s)`	$O(1)$	`len(L)`	$O(1)$	`len(S)`	$O(1)$
`in`	$O(n)$	`in`	$O(n)$	`in`	$O(1)$
`print(s)`	$O(n)$	`sorted(L)`	$O(nlogn)$		
`L1 += L2`	$O(len(L2))$				

```
def bigOH0(L): # L is a list, n = len(L)
    n = len(L)
    sortedL = sorted(L)
    count = 0
    for elem in L:
        count += 1
    return count
```

```
import copy
def bigOH1(S): # S is a set, n = len(S)
    n = len(S)
    i = 1
    while i < n:
        i *= 10
        if i in S:
            print("found")
    return None
```

```
def bigOH2(L): # L is a list, n = len(L)
    n = len(L)
    if 10 in L:
        if 10**2 in L:
            print("found")
    for i in range(len(L)):
        for j in range(len(L)):
            for k in range(len(L)):
                if i == j == k:
                    return False
    return True
```

```
import string
def bigOH3(s): # s is a string, n = len(s)
    n = len(s)
    mostRecent = None
    for c in s:
        if c in string.ascii_lowercase:
            mostRecent = c
    return mostRecent
```

```
def bigOH4(L): # L is a list, n = len(L)
    n = len(L)
    for i in range(0, n, n//4):
        print("woo")
    return None
```

```
def bigOH5(L): # L is a list, n = len(L)
    result = []
    for i in range(len(L)):
        result += L # careful, L is length N
    return sorted(result)
```

5. (12 points) **Free Response: mergeDictionaries(L)**

Write the function `mergeDictionaries(L)` that takes a list of dictionaries and merges them into a single dictionary. The merged dictionary should contain every key/value from each dictionary in L. Resolve ties between keys by mapping conflicting keys to the set of their values.

For example given:

```
L = [
{"a": 1, "b": 2},
{"a": 3, "c": 4},
{"e": 5}
]
```

mergeDictionaries(L) returns `{"a": set([1,3]), "b": 2, "c": 4, "e": 5}`.

Note: You assume that all values in the input dictionaries are immutable.

6. (15 points) **Free Response: Tasks, TaskList, PrioritizedTaskList**

Write the classes `Task`, `TaskList`, and `PrioritizedTaskList` so that the following test code runs without errors. Do not hardcode against the values used in the testcases, though you can assume the testcases cover the needed functionality. You must use proper object-oriented design, including **good inheritance**, or you will lose points!

```
# Tasks have a description, and a priority.
hw = Task("hw14", 4)
clean = Task("clean room", 3)

# You can compare tasks. Tasks are equal if they have the
# same description and priority.
assert(hw != clean)
assert(hw == Task("hw14", 4))
assert(hw != Task("hw14", 5))

# Tasks can be converted to strings
assert(str(hw) == "Task: hw14, Priority: 4")
assert(str(clean) == "Task: clean room, Priority: 3")

# You can create a task list. Task lists always start empty.
mondayList = TaskList()

# You can add task objects to task lists
mondayList.addTask(clean)
mondayList.addTask(hw)

# Getting the next task gives you the first task you entered
# and then removes that task from the task list.
assert(mondayList.getNextTask() == clean)
assert(mondayList.getNextTask() == hw)

# You can make a prioritized task list too!
netflix = Task("Netflix", 1)
betterMondayList = PrioritizedTaskList()
betterMondayList.addTask(netflix)
betterMondayList.addTask(clean)
betterMondayList.addTask(hw)

# Getting the next task gives you the task with the highest priority
# and then removes that task from the task list.
assert(betterMondayList.getNextTask() == hw)
assert(betterMondayList.getNextTask() == clean)
assert(betterMondayList.getNextTask() == netflix)
# Note: These test cases don't consider ties, you may ignore that case.
```

Additional Space for Answer to Question 6

Additional Space for Answer to Question 6

7. (15 points) **Free Response: getSeatingPlan**

For this problem you will write seating assignment software for a restaurant using recursive backtracking.

Specifically, write the function `getSeatingPlan(guestPreferences)` that takes a dictionary mapping groups of guests to a list of the tables that they could fit at and returns a dictionary mapping each group to a unique table.

The function should return None if there is no seating plan that has each group assigned to a unique table they could fit at.

For example given:

```
guestPreferences = {
    "group1": ["table1", "table3"],
    "group2":  ["table3"],
    "group3": ["table1", "table2", "table3"]
}
```

`getSeatingPlan(guestPreferences)` would return:

```
{
    "table1": "group1",
    "table2": "group3",
    "table3": "group2"
}
```

Hint: You may want to use a wrapper function.

Additional Space for Answer to Question 7

8. (20 points) **Free Response: optionallyOOpy Animation**

Using our animation framework and assuming run() is already written, write the init(data), keyPressed(event, data), mousePressed(event, data), timerFired(data), and redrawAll(canvas, data) functions for an animation which has the following elements:

1. Every time timerFired is called a black circle of radius 20 pixels appears on the screen in a random location.
2. Every time timerFired is called each circle on the screen moves right 30 pixels. When a circle's center travels past the width of the screen, the circle's center should reset to the left edge of the screen.
3. If the user clicks on a circle, that circle's radius increases by 10 pixels.
4. If the user presses the up arrow key, each circle on the screen should move up by 30 pixels. If the user presses the down arrow key, each circle on the screen should move down by 30 pixels. This movement should not have wrap around(the circles should just move off the screen).

Some notes/hints:

- Make reasonable assumptions for anything not specified here. We recommend that to save time writing you abbreviate canvas, event, and data. Use c, e and d, respectively.
- While we generally don't grade for style on exams, MVC violations are an exception. You will lose points if there are MVC violations in your code.
- You don't have to use OOP here, but it is recommended that you do.
- Hint: `random.randrange(0,10)` returns a random number between 0 and 10 (0 included, 10 excluded).

Additional Space 1 for Answer to Question 8

Additional Space 2 for Answer to Question 8