15-112
Fall 2018 Midterm 2
November 15, 2018

Name:

Andrew ID:

Recitation Section:

- You may not use any books, notes, extra paper, or electronic devices during this exam. There should be nothing on your desk or chair aside from this exam and any writing implements.
- You may not ask questions about the exam except for language clarifications.
- Show your work on the exam to receive credit.
- You may use the backs of pages as scratch paper. **Nothing written on the back of any pages will be graded.**
- You may complete the problems in any order you'd like; you may wish to start with the free response problems, which are worth most of the credit.
- All code samples run without crashing unless we state otherwise.
- You may assume that os, math, string, tkinter, random, and copy are imported; do not import any other modules.

Don't write anything in the table below.

Question	Points	Score
1	12	
2	12	
3	8	
4	10	
5	8	
6	10	
7	20	
8	20	
9	2	
Total:	102	

1. **Short Answer**

Answer each of the following questions. Read the instructions carefully! **For multiple choice questions, you MUST fill in the bubbles ENTIRELY.**

(a) (1 point) A filesystem is a naturally recursive data structure, and therefore has a base case and a recursive case in the data representation. Using **only one word** in each answer, what is a filesystem's:

Base case? _____

Recursive case? _____

(b) (2 points) Fill in the circles for **all** of the following statements which are correct according to our definitions of inheritance, classes, and instances.

- ◯ A class is like a template, while an instance is like a specific object.
- ◯ Dog could be a superclass of Animal.
- ◯ Different instances of a class have different methods.
- ◯ Shirt could be a subclass of Clothing.

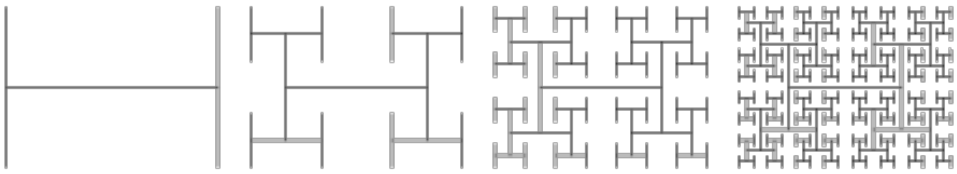
(c) (1 point) Assume that you're implementing side-scrolling as shown on the course website, with scrollX and scrollY (abbreviated here as sX and sY). If a circle is at position x,y with radius 5 in the main world map, how should it be drawn in redrawAll? Choose only **one** response.

- ◯ `canvas.create_oval(x-5, y-5, x+5, y+5)`
- ◯ `canvas.create_oval(x-5 - sX, y-5 - sY, x+5 - sX, y+5 - sY)`
- ◯ `canvas.create_oval(x-5 + sX, y-5 + sY, x+5 + sX, y+5 + sY)`
- ◯ `canvas.create_oval(sX - (x-5), sY - (y-5), sX - (x+5), sY - (y+5))`

(d) (2 points) Select **all** the valid reasons why you may want to use a wrapper function for a recursive problem:

- ◯ Separate out the recursive logic from any other logic to improve clarity of code
- ◯ It eliminates the need for recursion entirely
- ◯ It can allow you to initialize extra variables you might want to use for the recursive logic
- ◯ You shouldn't use wrapper functions - it is always better style to use default parameters instead

(e) (3 points) Consider the H-Fractal shown below. The leftmost figure is the fractal at depth 0. Assume that the base case of its recursive function draws one "H" shape.



(i) How many recursive calls to hFractal() are made in the recursive case? Choose only **one** response.

- ◯ 1
- ◯ 4
- ◯ 5
- ◯ None of the above

(ii) Select **all** the necessary inputs to the recursive function, based on our fractal demonstrations. Assume you may not use any input not on this list.

- ◯ X and Y coordinates for the center of the H
- ◯ Height of the H
- ◯ Number of H shapes that already exist
- ◯ Depth (or level)

(f) (3 points) The following three questions have to do with hash functions and sets. Choose only **one** answer for each.

(i) What is a hash function?

- ◯ A function that turns a mutable data type into an immutable data type
- ◯ A function that turns an immutable value into an integer
- ◯ A function that places an object into a set or dictionary
- ◯ A function that finds a value in a list in constant time

(ii) How is a hashed item put into a set?

- ◯ The hash value is used to create an index into a list, and the item is placed in an inner list at that index
- ◯ The hash value is placed into the set instead of the item
- ◯ The hash value is stored as a key and the item is stored as a value, just like in a dictionary
- ◯ The hash value and item are appended to the end of the set in a tuple

(iii) How is it possible to see if an item is in a set in O(1) time?

- ◯ Since a set has a constant size, we can scan the entire set in O(1) time
- ◯ We can hash the item again, and it's faster to look up the hash value than the original item, allowing us to search the set in O(1) time
- ◯ Hashing the item will tell us which index to look into, and we can search the inner list in O(1) time
- ◯ We can't see if an item is in a set in O(1) time

2. **Code Tracing**

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

(a) (4 points) CT 1

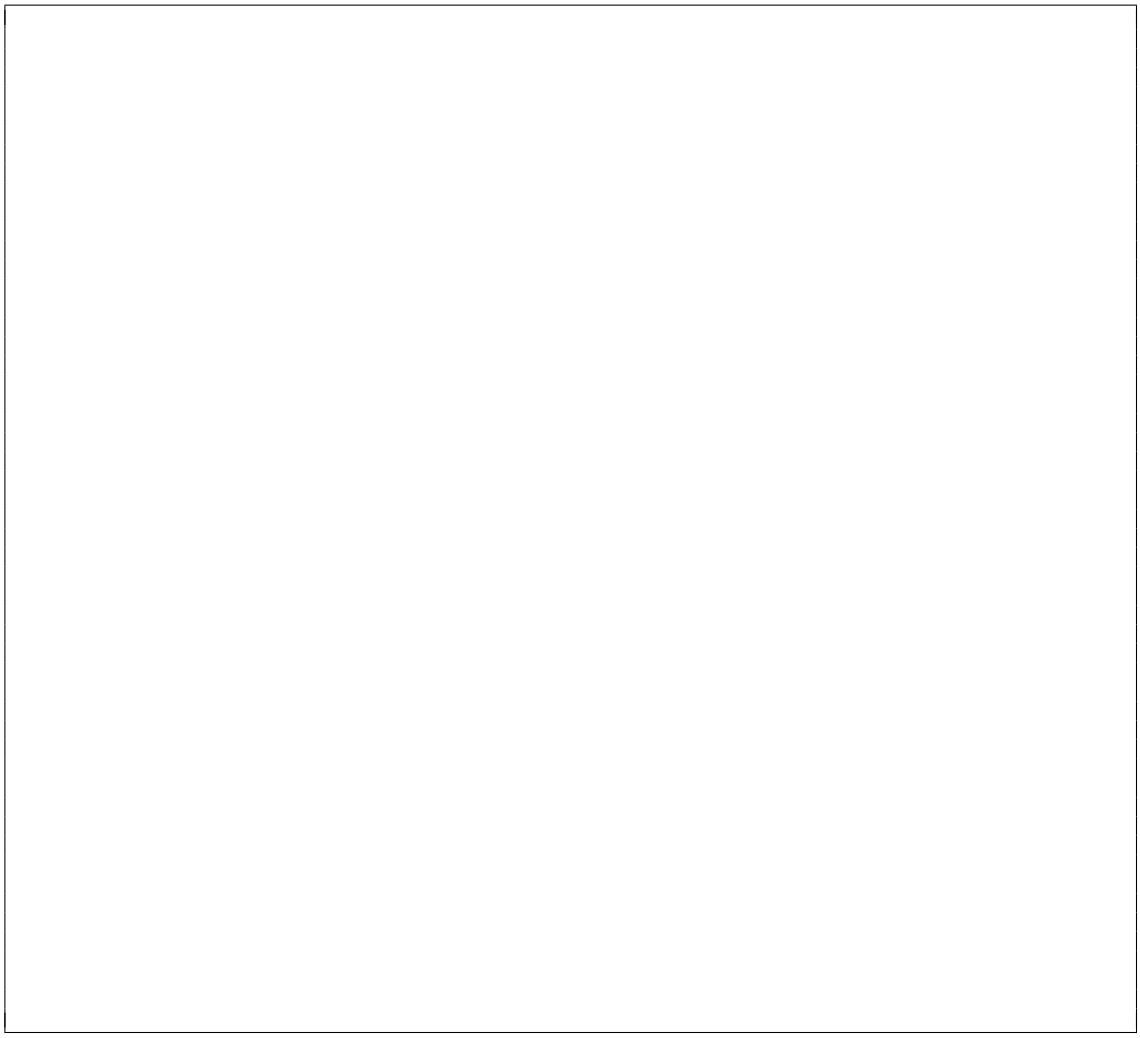
```
def ct1(s, lst):
    if s in lst:
        print(s, "repeat!")
        return
    else:
        lst.append(s)

    if len(s) == 1:
        print(s)
    else:
        if len(s) == 3:
            print("\n3:", s)
        ct1(s[1:], lst)
        ct1(s[:-1], lst)

ct1("woah", [])
```

(b) (4 points) CT 2

```
import copy
def ct2(lst):
    d={}
    for i in range(len(lst)//2):
        d[lst[i]]=lst[i*2]
    print(d)

    d[None]={0}
    for i in lst:
        if i not in d:
            d[None].add(i)
    print(d[None])

    b=copy.deepcopy(d)
    for i in b:
        if type(d[i]) not in {set, list, tuple}:
            if d[i] in d:
                del d[i]
    del d[None]
    print(d)

    return len(d)

lst=['x','x','y','z',4,5,6,6]
print(ct2(lst))
```

(c) (4 points) CT 3

```
def ct3(lst,x):
    L=0
    R=len(lst)-1
    while L<=R:
        i=(L+R)//2
        if lst[i]==x:
            return i
        elif lst[i]<x:
            L=i+1
        else:
            R=i-1
        print(lst[L:R+1])
    return -1

lst=[0,2,3,3,6,7,9,13,14,17,18]
print(ct3(lst,5))
```



3. **Reasoning Over Code**

For each function, find parameter values that will make the function return True. Place your answer (and nothing else) in the box below each block of code.

(a) (4 points) ROC 1: Find parameter values that will make the function roc1 return True. Place your answer (and nothing else) in the box below.

```
def roc1(lst):
    assert(isinstance(lst,list) and len(lst)==8)
    s=set()
    i=0
    while len(lst)>i:
        if lst[i] in s:
            i+=1
        else:
            s.add(lst.pop(i))

    assert(len(s)-len(lst)==2)

    for i in s:
        if i in lst:
            lst.remove(i)

    return(lst==["Z"])
```

(b) (4 points) ROC 2: Find parameter values that will make the function roc2 return True. Place your answer (and nothing else) in the box below.

```
def r1(x, y):
    if x<10:
        return [x,y]
    else:
        return r1(x%10,y+1)+r2(x//10,y+1)

def r2(x, y):
    if x<10:
        return [x,y]
    else:
        return r1(x//10,y+1)+r2(x%10,y+1)

def roc2(x):
    assert(isinstance(x,int))
    assert(100<x<999)
    return r1(x,x%10)==[5,6,3,7,1,7]
```

4. (10 points) **Big-O:** For each function shown below, write next to each line of the function either the Big-O runtime of the line or the number of times the line loops. Then write the total Big-O runtime of the function in terms of N in the box to the right of the code. **All answers must be simplified- do not include lower-order terms! For full credit, you must include line-by-line Big-O.**

Built-in Big-O Runtimes			
General		Strings	
`len(item)`	$O(1)$	`chr(s) / ord(s)`	$O(1)$
`item[i]`	$O(1)$	`s.count(c)`	$O(N)$
`c in item` (for strings and lists)	$O(N)$	`s.find(c)`	$O(N)$

```
 1: def bigO1(L): # L is a list of length N     # Big-O
 2:     x = y = z = 0                           #______
 3:     for i in range(len(L)//2):              #______
 4:         x += L[i]                           #______
 5:     for j in range(0, len(L), len(L)//2):   #______
 6:         y += L[j]                           #______
 7:     k = len(L) - 1                          #______
 8:     while k > 0:                            #______
 9:         z += L[k]                           #______
10:         k = k // 2                          #______
11:     return (x - y) / z                      #______
```

```
 1: import string
 2: def bigO2(s): # s is a string of length N   # Big-O
 3:     x = 0                                   #______
 4:     for c in s:                             #______
 5:         nextC = chr(ord(c) + 1)             #______
 6:         if c in string.digits:              #______
 7:             x += s.count(c)                 #______
 8:         elif nextC in s:                    #______
 9:             x -= s.find(nextC)              #______
10:     return x                                #______
```

```
 1: def bigOh3(L): # L is a list of length N    # Big-O
 2:     x = 0                                   #______
 3:     n = len(L)                              #______
 4:     for i in range(n):                      #______
 5:         if i > n/2:                         #______
 6:             for j in range(n**2):           #______
 7:                 if i == j and i in L:       #______
 8:                     x += 1                  #______
 9:     return x                                #______
```

5. (8 points) **Free Response: getItemCounts**

   Write the function `getItemCounts(lst)` which takes a list of values and returns a dictionary mapping each value in the list to the number of times it appears. For example,

   ```
   getItemCounts(["a", "b", "c", "a", "a", "c"])
   ```

   would return

   ```
   {"a" : 3, "b" : 1, "c" : 2}
   ```

   Your solution must use recursion. If you use any loops, comprehensions, or iterative functions, you will receive no points on this problem.

6. (10 points) **Free Response: Circle Animation**

Assuming the `run()` function is already written for you, write the `timerFired(d)` function for the following animation. `init(d)` and `redrawAll(c, d)` have already been implemented for you; the code is included below. Do not modify these functions.

1. Every second, a circle is added to the screen. The circle should be placed in a random location on the screen and should have a radius of 20. The number 0 should be displayed in the middle of the circle.
2. All circles on the screen move to the right continuously. As soon as a circle moves entirely off the screen, it reappears on the left side, still moving to the right. Every time a circle wraps around the screen in this way, the number in the middle of the circle should increase by one.

Make reasonable assumptions for anything not specified here. Do not hardcode values for the width or height. We recommend that, to save time writing, you abbreviate canvas, event, and data: use c, e and d, respectively. You should use short variable names.

```
# Starter code
def init(d):
    d.timerDelay = 100
    d.timePassed = 0
    d.circles = [ ]

def redrawAll(c, d):
    for circle in d.circles:
        x, y, count = circle
        c.create_oval(x - 20, y - 20, x + 20, y + 20)
        c.create_text(x, y, text=str(count))
```

Additional Space for Answer to Question 6

7. (20 points) **Free Response: OOP**

Write a set of classes so that they pass the following test cases. Your classes should include Bird and Chicken, but are not limited to these two. You may not hardcode any test cases. For full credit you must use inheritance appropriately.

```
###Make a bird named Steve
b = Bird("Steve",5)
assert(b.name=="Steve")
assert(b.size==5)
assert(b.fly()=="can fly")
#Feed Steve
b.feed(2)
assert(b.size==7)
assert(repr(b)=="Steve weighs 7oz and can fly")
#Birds can't fly if their size is twice as big as their original size!
b.feed(4)
assert(b.size==11)
assert(b.fly()=="can't fly")
#Birds are equal if they have the same name and size
b2=Bird("Steve",11)
assert(b==b2)

###Make a chicken named Betty
c = Chicken("Betty")
assert(isinstance(c,Bird))
assert(c.name=="Betty")
assert(c.size==20)
#Chickens can't fly
assert(c.fly()=="can't fly")
assert(repr(c)=="Betty weighs 20oz and can't fly")
#You can still feed chickens though
c.feed(5)
assert(c.size==25)

###Chickens can also lay chicken eggs
e=c.layChickenEgg()
assert(type(e)==ChickenEgg)
assert(isinstance(e,Bird)==False)
###Other birds can't lay chicken eggs though
try:
    b.layChickenEgg()
    eggLaid=True
except:
    eggLaid=False
assert(not eggLaid)
```

Additional Space for Answer to Question 7

Additional Space for Answer to Question 7

8. (20 points) **Free Response: packItems**

You're about to travel home for Thanksgiving, so you need to pack up your stuff! However, you have so much stuff to bring home that you'll need to use multiple bags. This is made more difficult by the fact that each item has a weight, and each bag has a maximum weight that it can handle- if a bag gets overloaded, it will break. Your task is to find a way to organize all of your items into all of your bags such that no item is left behind and no bag contains a heavier weight than its limit.

Solve this problem by writing the function `packItems(items, bagSizes)`. The function takes two lists: `items`, which is a list of the weights of all the items, and `bagSizes`, which is a list of the weight limits of all the bags. It should return a list of "bags", where a bag is a list of items (numbers). The bag list should be the same length as `bagSizes`, and each bag should not weigh more than the corresponding weight limit.

For example, say you have two bags with weight limits of 12 and 9, and the following list of item weights: `[4, 8, 1, 4, 3]`. The function call for this would be

```
packItems([4, 8, 1, 4, 3], [12, 9])
```

and it would return

```
[ [4, 8], [1, 4, 3] ]
```

Note that the first bag sums to 12 (the first weight limit) while the second sums to 8 (less than the second weight limit). There are other possible packings for this set of items; any valid packing is acceptable.

If the provided bag sizes were instead `[10, 10]`, there would be no valid way to pack the bags; in that case, the function should return `None` instead of a list of bags. You are guaranteed that item weights and bag sizes will be non-negative.

Note: You must use recursion and backtracking to solve the problem to receive credit, even if another approach would work.

Additional Space for Answer to Question 8

Additional Space for Answer to Question 8

9. (2 points) **Bonus CT:** Only try this if you're done with the other questions and are bored! Print what the following code prints

```
def bonusCt():
    def f(x,y=[]): return y if x < 2 else y.append(x % (x - 5)) or g(y)
    def g(z): return f(14-sum(z)*len(z)) if len(z) <= 2  \
                    else f(sum([abs(x) for x in z]) % 10)
    return eval("".join([str(x) for x in f(7)]))

print(bonusCt())
```