15-112
Spring 2021 Exam 2
November 9, 2021

Name:

Andrew ID:

- 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 use the backs of pages as scratch paper. Nothing written on the back of any pages will be graded.
- All code samples run without crashing. Assume any imports are already included as required.
- You may assume that random, math, string, basic_graphics, cmu_112_graphics and copy are imported; do not import any other modules.

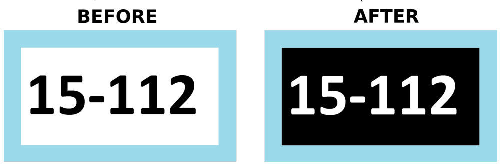
Don't write anything in the table below.

Question	Points	Score
1	12	
2	15	
3	10	
4	15	
5	20	
6	15	
7	20	
Total:	107	

There are 107 points on this exam, but your final score will be out of 100. (So, 7 points are effectively bonus.) However, the highest score than can be received on this exam is still 100.

1. Short Answer / Multiple Choice

   (a) (3 points) Which one of the following functions produces the image manipulation shown below. You can assume that this is called within the appropriate image processing helper code. *Circle the letter of the correct answer (A, B, C, or D).*



   A. `f1()`
   B. `f2()`
   C. `f3()`
   D. `f4()`

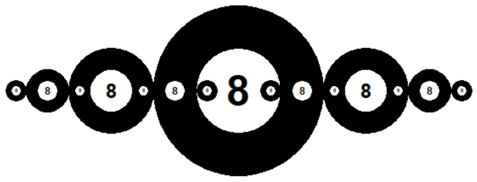
```
def f1():
    for i in range(width):
        for j in range(height):
            color = ImageWriter.getColor(mypic, i, j)
            newColor = [color[2],color[0],color[1]]
            ImageWriter.setColor(mypic,i,j,newColor)

def f2():
    for i in range(width):
        for j in range(height):
            color = ImageWriter.getColor(mypic, i, j)
            avg = sum(color)//3
            if avg > 75:
                newColor = [255,255,255]
            else:
                newColor = [0,0,0]
            ImageWriter.setColor(mypic,i,j,newColor)

def f3():
    for i in range(width):
        for j in range(height):
            color = ImageWriter.getColor(mypic, i, j)
            newColor=color
            if color == [0,0,0]:
                newColor = [255,255,255]
            elif color == [255,255,255]:
                newColor = [0,0,0]
            ImageWriter.setColor(mypic,i,j,newColor)

def f4():
    for i in range(width):
        for j in range(height):
            color = ImageWriter.getColor(mypic, i, j)
            avg = sum(color)//3
            newColor=[avg,avg,avg]
            ImageWriter.setColor(mypic,i,j,newColor)
```

(b) (3 points) Which one of the following functions produces the fractal pattern shown below. You can assume that this is called within the appropriate graphics helper code. The function is called with the following arguments: `(canvas, width//2, height//2, 100, 4)`. *Circle the letter of the correct answer (A, B, C, or D).*



A. `f1()`
B. `f2()`
C. `f3()`
D. `f4()`

```
def f1(cv, x, y, r, depth):
    if depth>0:
        cv.create_oval(x-r,y-r,x+r,y+r,fill="black")
        cv.create_oval(x-r//2,y-r//2,x+r//2,y+r//2,fill="white")
        cv.create_text(x,y,text="8",fill="black", font=f'arial {r//4} bold')
        f1(cv, x+1.5*r,y,r//2,depth-1)
        f1(cv, x-1.5*r,y,r//2,depth-1)

def f2(cv, x, y, r, depth):
    if depth==0:
        return
    cv.create_oval(x-r,y-r,x+r,y+r,fill="black")
    cv.create_oval(x-r//2,y-r//2,x+r//2,y+r//2,fill="white")
    cv.create_text(x,y,text="8",fill="black", font=f'arial {r//4} bold')
    f2(cv, x+r,y,r//2,depth-1)
    f2(cv, x-r,y,r//2,depth-1)

def f3(cv, x, y, r, depth):
    if depth>0:
        cv.create_oval(x-r,y-r,x+r,y+r,fill="black")
        cv.create_oval(x-r//2,y-r//2,x+r//2,y+r//2,fill="white")
        cv.create_text(x,y,text="8",fill="black")
        f3(cv, x+r,y,r//2,depth-1)
        f3(cv, x-r,y,r//2,depth-1)

def f4(cv, x, y, r, depth):
    if depth==0:
        return
    cv.create_oval(x-r,y-r,x+r,y+r,fill="black")
    cv.create_oval(x-r//2,y-r//2,x+r//2,y+r//2,fill="white")
    cv.create_text(x,y,text="8",fill="black", font=f'arial {r//4} bold')
    f4(cv, x+2*r,y,r//2,depth-1)
    f4(cv, x-2*r,y,r//2,depth-1)
```

(c) (3 points) Give an example of an MVC violation.

(d) (3 points) Consider the following function. Fill in a **single line of code** to complete it. (If you can't do it in one line, you can use more than one line for, at most, half credit.)

```
# Return a dictionary mapping words in L to the number of times they occur in L
# For example, wordCounter(["a","b","b","c","a","monkey"]) returns
# {'a': 2, 'b': 2, 'c': 1, 'monkey': 1}
def wordCounter(L):
    d = {}
    for word in L:

          ______________________________________________________________
    return d
```

2. **Code Tracing**

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

   (a) (5 points) CT1

```
def f(u):
    if 8 in u:
        print(f'8: {u[8]}')
        del u[8]
    return u

def ct(L):
    s = set(L)
    d = dict()
    for v in L:
        d[v] = d.get(v,v) + min(s)
        s.add(d[v])
    u = f(d)
    print(f's = {s}')
    print(f'd = {d}')
    print(f'u = {u}')

ct([8,4,8,4,2])
```

(b) (5 points) CT2

```
class A(object):
    def __init__(self):
        self.b = "Puppy"

    def f(self):
        print("Dog")

    def g(self):
        print("Cat")

    def h(self):
        print(f"{self.b}: Pony")
        self.g()

class B(A):
    def __init__(self, s):
        self.b = s
        super().__init__()

    def g(self):
        print("Falcon")
        super().g()

    def h(self, s):
        super().h()
        print(f"{s}: Tiger")

def ct2():
    b = B("Sheep")
    b.h("Moo")

ct2()
```

(c) (5 points) CT3

```
def ct3(L, d=0):
    print(f"IN L:{L} d:{d}")
    if len(L) > 1:
        m = len(L)//2
        ct3(L[m:], d+1)
        ct3(L[:m], d+1)
    print(f"OUT L:{L} d:{d}")

ct3([1,2,3])
```

3. **Reasoning Over Code**

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

   (a) (5 points) ROC1

```
def roc1(d):
    assert(len(d) == 4)
    for item in d:
        assert(isinstance(item, str) and isinstance(d[item],str))
        if d[item] == item or d[item] not in d:
            return False
    return True
```

   (b) (5 points) ROC2

```
def f(L, depth=2):
    if len(L) < 2:
        return depth==0
    assert(len(set(L)) == len(L))
    assert(len(L)%2 == 0)
    assert(L[0] < L[-1])
    assert(L[0] + L[-1] < 6)
    m = len(L)//2
    a = f(L[:m], depth - 1)
    b = f(L[m:], depth - 1)
    return a and b

def roc(L):
    return f(L)
```

4. (15 points) **Free Response: Recursive Palindromes**

   A palindrome is a word that is spelled the same forwards and backwards. Here are some examples of palindromes: racecar, tacocat, anna

   Write the function `isPalindrome(word)` that returns `True` if the string `word` is a palindrome and `False` otherwise.

   Your solution must use recursion. If you use any loops, comprehensions, or iterative functions, you will receive no points on this problem. This restriction includes slicing with a negative step (so you may not use `s[::-1]`, for example.) Regular slicing (`s[2:4]`, for example) is fine, however.

5. (20 points) **Free Response: OOP (Medium)** Consider the following code designed to test two different classes: Book and Journal:

```
# A book has a title, authors, and publication year
# It also has a number of copies in the library,
# and the number of copies currently available
b1 = Book("The Art of Computer Programming",
              "Donald Knuth", 1968, 5)
assert(str(b1) == """Donald Knuth, "The Art of Computer Programming", 1968, 5/5""")
# You can loan a book, as long as there are copies available
b1.loan()
assert(b1.getCopiesAvailable() == 4)
b1.loan()
assert(b1.getCopiesAvailable() == 3)
for i in range(3):
    b1.loan()
assert(str(b1) == """Donald Knuth, "The Art of Computer Programming", 1968, 0/5""")
try: b1.loan() # no more copies available, should fail!
except: ok = True
assert(ok)
# You can return a library item
b1.returnCopy()
assert(b1.getCopiesAvailable() == 1)
b1.returnCopy()
assert(b1.getCopiesAvailable() == 2)
# if more copies than the initial number of copies are returned, they are discarded
for i in range(10):
    b1.returnCopy()
assert(b1.getCopiesAvailable() == 5)
# When determining equality, book titles are not case sensitive, but author
# is. Also, the number of copies doesn't matter.
b2 = Book("The C Programming Language",  "Kernighan and Ritchie", 1978, 10)
b3 = Book("the c programming language",  "Kernighan and Ritchie", 1978, 8)
b4 = Book("THE C PROGRAMMING LANGUAGE",  "Kernighan and Ritchie", 1978, 10)
b5 = Book("THE C PROGRAMMING LANGUAGE",  "kernighan and Ritchie", 1978, 10)
b6 = Book("The C Programming Language",  "Kernighan and Ritchie", 1988, 10)
assert(b2 == b3)
assert(b2 == b4)
assert(b4 != b5)
assert(b2 != b6)
# You should be able to hash books. The rules follow those for equality.
collection = set()
collection.add(b1)
collection.add(b2)
assert(b1 in collection)
assert(b2 in collection)
assert(b3 in collection)
assert(b4 in collection)
assert(b5 not in collection)
assert(b6 not in collection)
# A Journal is a book with various authors
j1 = Journal("Science Vol 374 Issue 6568", 2021, 8)
assert(str(j1) == """Various Authors, "Science Vol 374 Issue 6568", 2021, 8/8""")
assert(j1 != b1)
assert(isinstance(j1, Book))
```

Write the classes Book and Journal so that the test code runs as specified. Do not hardcode against the values used in the testcases, though you can assume the testcases cover the needed functionality. For full credit, you must use proper object-oriented design, including good inheritance and avoiding unnecessary code duplication.

Additional Answer Space for Question 5

6. (15 points) **Free Response: Most Frequent Visitor**

   Write the function `mostVisits(logbook)` that is given a dictionary mapping days of the week to the list of students who visited CMU-Q on that day, and returns a set that contains the student (or students, if there is a tie) who visited on the most number of days that week. There is one caveat: The log system might register a visit multiple times on the same day, therefore one name might appear multiple times in a list, but it should be counted only once per day.

   For example, given the dictionary:

```
{ "Sunday" : [ "Layla", "Peter", "Otto", "Amir" ],
"Monday" : [ "Yusuf", "Layla", "Bernard", "John" ],
"Tuesday" : [ "Yusuf", "Peter", "Otto", "Layla", "Salma", "Otto" ],
"Wednesday" : [ "Otto", "Layla", "Yusuf", "Otto" ] }
```

   The function should return the set `{"Layla"}`, since Layla visited the CMU-Q building four days (Otto entered the building more times, but only visited on three different days).

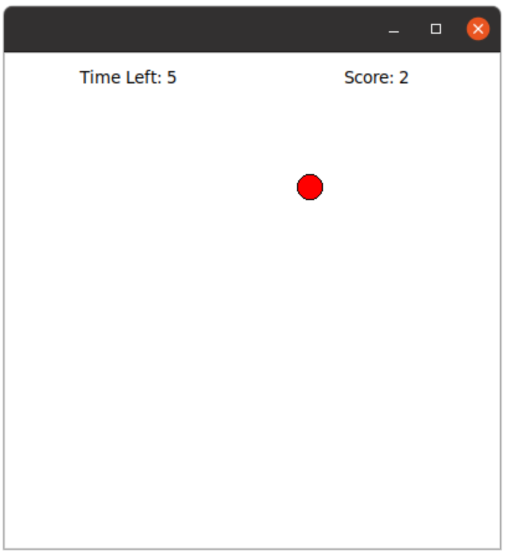
   If Layla had not visited the building on Monday, then it would return `{"Layla", "Otto", "Yusuf"}`, since each student would have visited exactly three days.

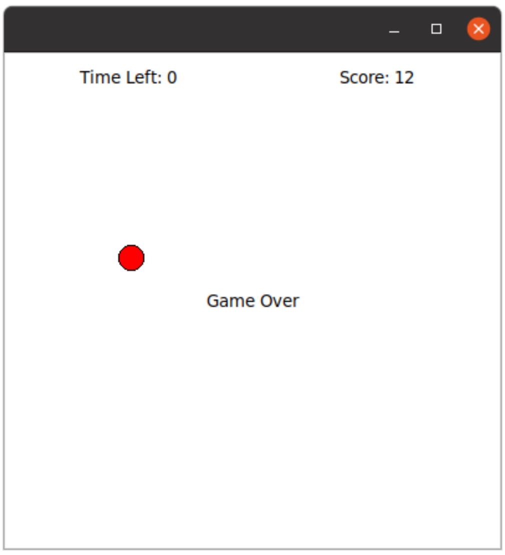
7. (20 points) **Free Response: Clicky Ball**

   Write `appStarted`, `keyPressed`, `mousePressed`, `redrawAll`, and `timerFired` that implement the game described below.

   1. The game begins with a red ball (radius of 10 pixels) placed in a random location on the screen.
   2. In the upper left hand corner of the screen is a timer. It counts down from 10 seconds to 0 seconds, only displaying whole seconds.
   3. In the upper right hand corner of the screen is the user's current score.
   4. When the user clicks on the ball, one point is added to the score and the ball "teleports" to a new, random location.
   5. After the 10 second timer has counted down to 0, the game stops and displays "Game Over". When the game is stopped, the time left stays at 0 and the user cannot get additional points.
   6. At any time, the user can press "r" to reset the game and play again.

   Here are some screen shots. The first was taken during the game, the second was taken after the game was over.





Answer space for Question 7

Answer space for Question 7