Dartmouth College
Computer Science 1, Spring 2014
Exam 2

Thursday, May 8

Print your name: SOLUTION

Circle your recitation section leader's name.

Xiaole (Caleb) An	Eric Chen	Tyler Crowe	Max Deibel
Derek DeWitt	Lotanna Ezenwa	Emily Greene	Nicole Hedley
Naho Kitade	Anna Knowles	Matthew Krantz	Jai Lakhanpal
Joshua Lang	Justine Lee	Christopher Leech	Randy Li
Bridget Melvin	Taylor Neely	Mehdi Oulmakki	Chander Ramesh
Arun Reddy	Benjamin Ribovich	Benjamin Rush	Nicholas Schwartz
Kameko Winborn	Garrett Watumull	Guanyang (Ethan) Yu	

- If you need more space to answer a question than we give you, you may use the additional blank sheet of paper attached to your exam. Make sure that we know where to look for your answer.
- Read each question carefully and make sure that you answer everything asked for. Write legibly so that we can read your solutions. Please do not write anything in red.
- We suggest that for solutions that require you to write Python code, you include comments. They will help your grader understand what you intend, which can help you get partial credit.
- You have until 9:00 pm to complete this exam.
- You do not need to submit your crib sheet.

Question	Value	Score
1	10	
2	20	
3	10	
4	15	
5	5	
6	10	
7	10	
8	10	
9	10	
Total	100	

Question 1 10 points

For each part of the question, answer True or False. Use the space below each question for your answer.

(a) (2 points)
In a class constructor, `self` is the address of an object.

TRUE

(b) (2 points)
Besides the class constructor, every class must have at least one member function.

FALSE

(c) (2 points)
$O(\sqrt{n}) > O(n \log(n))$.

~~TR~~ FALSE

(d) (2 points)
If a recursive function does not have a base case it is likely to result in an error.

TRUE

(e) (2 points)
In the best case, linear search has the same runtime as binary search.

TRUE

Question 2 20 points

For each part of the question, I will provide some Python code. Tell me what will be printed on the console. If you think that there is an error in the code that would prevent output, briefly describe the error. Use the space to the right of each question for your answer.

(a) (2 points)

```
x = 0
for i in range(0,3):
    for j in range(0,3):
        x = x + 1
print x
```

9

(b) (2 points)

```
def do(a,b=False):
    if b:
        return a
    return -1 * a

print do(1) + do(1,True)
```

0

(c) (2 points)

```
L = []
L.append([])
L[0].append(1)
L.append(2)
print L
```

[[1], 2]

(d) (2 points)

```
L = [[3,2,1], [6,5,4]]
for i in range(0,2):
    L[i].sort()
print L
```

[[1,2,3], [4,5,6]]

(e) (2 points)

```
if( False ):
    print "here1"
elif( True ):
    print "here2"
elif( True ):
    print "here3"
else:
    print "here4"
```

HERE2

(f) (2 points)

```
# this code is in ball_driver.py
from ball import *
b = Ball(10,10,10)
for i in range(0,5):
    b.modify(1)
print b

# ---------------------------
# this code is in ball.py
class Ball:
    def __init__(self,x,y,r):
        self.x = x
        self.y = y
        self.r = r
    def modify(self, z):
        self.r = self.r + z
    def __str__(self):
        return str(self.x) + ", " + str(self.y) + ", " + str(self.r)
```

10, 10, 15

(g) (2 points)

```
# this code is in mystery_driver.py
from mystery import *
X = Mystery("abc")
Y = X
X.name = "xyz"
print X.name, Y.name

# ---------------------------------
# this code is in mystery.py
class Mystery:
    def __init__(self,name):
        self.name = name
```

XYZ XYZ

(h) (2 points)

```
def do(x):
    if( x == 0 ):
        return 0
    else:
        return x + do(x-1)
print do(5)
```

15

(i) (2 points)

```
def do(x):
    return do(x-1)
print do(1)
```

ERROR

(j) (2 points)

```
def r(n):
    print n
    if( n <= 10 ):
        return n * 2
    else:
        return r(r(n/3))
print r(12)
```

12
4
8
16

Question 3 10 points

For each part of the question I will provide a Python function. Using big-Oh notation, give the running time of each function.

(a) (2 points)

```
def func(n):
   for i in range(1,n):
      for j in range(1,2*n):
         for k in range(1,3*n):
            print i
```

$O(n^3)$

(b) (2 points)

```
def func(n):
   i = 1
   while i < n-10000:
      for j in range(0,1000000):
         print j
      for k in range(0,100000000000):
         print k
      i = i + 1
```

$O(n)$

(c) (2 points)

```
def func(n):
   sum = 0
   for i in range(0,sqrt(n))
      sum = sum + i
   return sum
```

$O(\sqrt{n})$

(d) (2 points)

```
def func(n):
   while n > 1:
      n = n / 10
```

$O(\log(n))$

(e) (2 points)

```
def func(n):
   print n
   print n*n
   print n*n*n
   print n*n*n*n
```

$O(1)$

Question 4 15 points

Recall the `Eye` and `Face` classes that we discussed in lecture (to refresh your memory, they are provided on the next page). Write a `Crowd` class that uses these two classes to create a sea of faces that each track your mouse movement.

Your constructor should take as a parameter the number of faces in the crowd and initialize two instance variables: (1) the number of faces `N`; and (2) a list of faces, `the_crowd`, each of size 50 pixels and placed at a random position in the window (it is fine if the faces occlude one another or are not fully visible in the window).

Your `Crowd` class should have three functions: (1) a constructor; (2) the function `lookat`; and the function `draw`. The function `lookat` should take as input a `x,y` location and have each face in the crowd look at this position. The function `draw` should draw each face in the crowd. Your functions cannot directly call any `Eye` member functions, but should call the appropriate `Face` member functions `lookat` and `draw`.

Use the space below and the next page (if necessary) for your solution.

```
FROM CS1LIB IMPORT *
FROM FACE IMPORT FACE

FROM RANDOM IMPORT *

CLASS CROWD:
    DEF __INIT__(SELF, N):
        SELF.N = N
        SELF.THE_CROWD = []
        FOR i IN RANGE(0, N):
            SELF.THE_CROWD.APPEND(
                          (FACE(x, y, 50))
            x = UNIFORM (20, 380)
            y = UNIFORM (20, 380)
```

```
# THIS PAGE LEFT INTENTIONALLY BLANK
```

```
DEF LOOKAT(SELF, x, y):
    FOR i IN RANGE(0, SELF.N):
        SELF.THE-CROWD[i].LOOKAT(x,y)


DEF DRAW(SELF):
    FOR i IN RANGE(0, SELF.N):
        SELF.THE-CROWD[i].DRAW()
```

```
# ----- EYE CLASS -----
from cs1lib import *
from math import *

class Eye:
    def __init__(self, x, y, rad, r = 0, g = 0, b = 1):
        # eye position and size
        self.x = x
        self.y = y
        self.rad = rad
        self.direction = 0

        # eye color
        self.r = r
        self.g = g
        self.b = b

    def lookat(self, lx, ly):
        self.direction = atan2(ly-self.y, lx-self.x)

    def draw(self):
        # draw outer circle
        enable_stroke()
        set_fill_color(1,1,1)
        draw_circle(self.x, self.y, self.rad)

        # draw inner circle
        set_fill_color(self.r, self.g, self.b)
        ix = 0.4 * self.rad * cos(self.direction) + self.x
        iy = 0.4 * self.rad * sin(self.direction) + self.y
        draw_circle(ix, iy, 0.5*self.rad)

# ----- FACE CLASS -----
from cs1lib import *
from eye import Eye

class Face:
    def __init__(self, x, y, size):
        # face position and size
        self.x = x
        self.y = y
        self.size = size

        # create eyes
        self.lefteye = Eye( x-15, y-15, 10 )
        self.righteye = Eye( x+15, y-15, 10 )

    def lookat(self, lx, ly):
        self.lefteye.lookat( lx, ly )
        self.righteye.lookat( lx, ly )

    def draw(self):
        enable_stroke()
        set_fill_color(1,1,1)
        draw_circle(self.x, self.y, self.size) # draw face
        self.lefteye.draw() # draw eyes
        self.righteye.draw() # draw eyes
        draw_line( self.x, self.y, self.x, self.y+15 ) # draw nose
        draw_line( self.x-15, self.y+20, self.x+15, self.y+20 ) # draw mouth
```

Question 5 5 points

Create a driver that tests your `Crowd` class. This Python code should create a crowd with 20 faces and then as long as the graphics window is open it should have each face look at your mouse position (as we did in lecture with the `Face` class). This function cannot directly call any `Eye` or `Face` member functions.

```
FROM CS1LIB IMPORT *
FROM CROWD IMPORT CROWD

DEF MAIN():
    ENABLE_SMOOTHING()  # OPTIONAL
    SET_CLEAR(1, 1, 1)  # OPTIONAL
    C = CROWD(20)
    WHILE NOT WINDOW_CLOSED():
        CLEAR()
        CROWD.LOOKAT(MOUSE_X(), \
                     MOUSE_Y())
        CROWD.DRAW()
        REQUEST_REDRAW()
        SLEEP(0.02)

START_GRAPHICS(MAIN)
```

Question 6 10 points

Bubble sort is a simple sorting algorithm that works by repeatedly stepping through a list to be sorted, comparing each pair of adjacent items and swapping them if they are in the wrong order. Below is most of the code for bubble sort. Complete this code for bubble sort by adding a few lines of code where noted below.

```
def bubble_sort(A):
    n = len(A)
    for i in range(0,n):
        for j in range (i+1,n):
            if A[i]>A[j]:
                # add your code here.
```

```
TEMP = A[i]
  A[i] = A[j]
  A[j] = TEMP
```

Question 7 10 points

Write a recursive function `count` that takes as a parameter a list and returns the number of negative numbers in the list. Your function must recursively compute the number of negative numbers and cannot use a `while` or `for` loop. You can assume that the list passed to the function will always contain numbers.

```
DEF  COUNT (L):
    IF  L == []:
        RETURN 0
    ELIF  L[0] < 0:
        RETURN  1 + COUNT(L[1:])
    ELSE
        RETURN  COUNT(L[1:])
```

Question 8 10 points

(a) (4 points)

Below is my solution to `binary_search` (short assignment 7). You decide that you don't like the idea of splitting the list exactly in half on each recursive call. Briefly explain how you would modify this function to split the list at a random location on each recursive call.

```
def binary_search(the_list, key, left=None, right=None):
    if left == None and right == None:
        left = 0
        right = len(the_list) - 1

    if left > right:
        return None
    else:
        midpoint = (left + right) / 2
        if the_list[midpoint] == key:
            return midpoint
        elif the_list[midpoint] > key:
            return binary_search(the_list, key, left, midpoint-1)
        else:
            return binary_search(the_list, key, midpoint+1, right)
```

$\rightarrow$ MIDPOINT = RANDINT (LEFT, RIGHT)

(b) (6 points)

Using big-O notation, what is the best and worst case runtime for your modified search algorithm?

BEST: O(1)

WORST: O(n)

Question 9 10 points

Consider the following recursive function.

```
def mystery( a, b ):
   if( b < 0 ):
      return -1 * mystery(a,-b)
   elif( b == 0 ):
      return 0
   elif( b == 5 ):
      return 1
   else:
      return b + mystery(a,b+1)

print mystery( 5, -3 )
```

For each call to the function `mystery`, fill in the appropriate values for `a`, `b`, `return to`, and `return value` in the call stack below. Then, specify the order of pushes and pops of each stack frame onto the call stack. For example, your answer might look something like: *push ?, push ?, pop ?, push ?*, ..., where *?* is replaced with the numeric value to the left of each frame.

1	*function:* **mystery**	**a:** 5 **b:** -3 *return to:* MAIN	*return value:* -8	
2	*function:* **mystery**	**a:** 5 **b:** 3 *return to:* MYSTERY	*return value:* 8	
3	*function:* **mystery**	**a:** 5 **b:** 4 *return to:* MYSTERY	*return value:* ~~10~~	5
4	*function:* **mystery**	**a:** 5 **b:** 5 *return to:* MYSTERY	*return value:* ~~6~~	1

PUSH 1
PUSH 2
PUSH 3
PUSH 4

POP 4
POP 3
POP 2
POP 1

```
# THIS PAGE LEFT INTENTIONALLY BLANK
```