

CS111 EXAM 1

March 05, 2015

YOUR NAME*: _____

*by writing your name above you are stating that you abided by the course policies while taking this exam

Please indicate your lecture by checking the appropriate box (so we can return your exam to you):

Lyn 9:50am

Rhys 11:10am

Rhys 1:30pm

This exam has 5 problems. Some problems have several parts. The number of points for each problem is shown in square brackets next to the problem or part. There are 100 total points on the exam.

Write all your answers on the exam itself.

The exam is open book. You may refer to your notes, and other course materials **except that you may not use another person's notes, or any materials from prior semesters of CS111. You may not access any electronic device at any time.**

Please keep in mind the following tips:

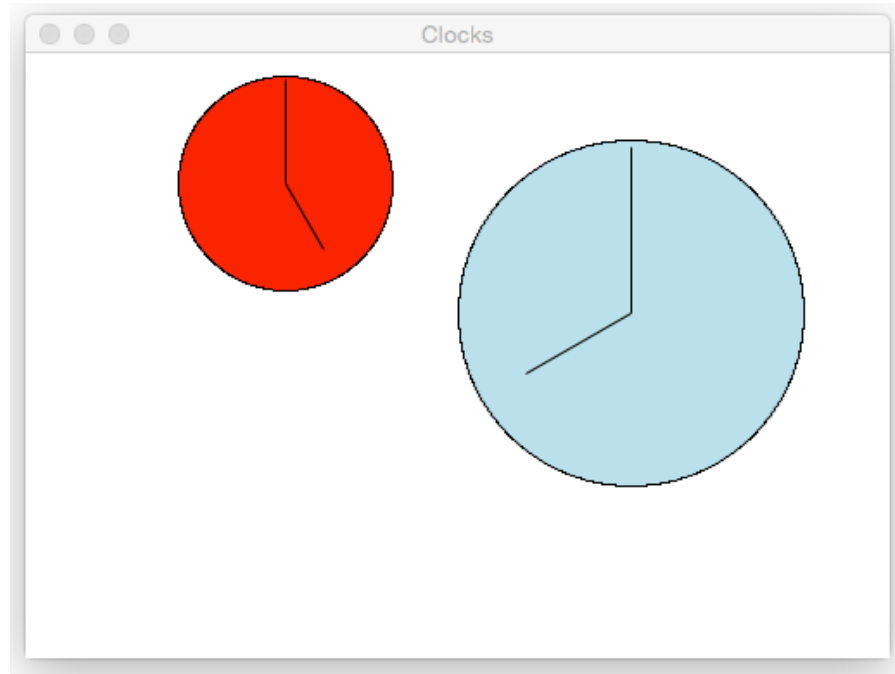
- *First skim through the entire exam.* Work first on the problems on which you feel most confident. You do not need to do the problems in the order they are presented.
- *Try to do something on every problem* so that you can receive partial credit. For programming problems, you can receive partial credit for explaining your strategy with words and pictures.
- *Allocate your time carefully.* If you are taking too long on a problem, wrap it up and move on.

The following table will be used in grading your exam:

Problem	Score
Problem 1 [21 pts]	
Problem 2 [12 pts]	
Problem 3 [16 pts]	
Problem 4 [12 pts]	
Problem 5 [39 pts]	
Total [100 pts]	

Problem 1: Capture the Pattern [21 points]

I must govern the clock, not be governed by it – Golda Meir.



Using `cs1graphics`, the canvas above can be produced with the code below:

```
from cs1graphics import *

wall = Canvas(500,350,'white','Clocks')

# Draw first clock, red, diameter 125, time 5pm, at (150,75):
clock1 = Layer()
clockface1 = Circle(125/2.0) # 125 is clockface diameter
clockface1.setFillColor('red')
clock1.add(clockface1)
minutehand1 = Path(Point(0,0), Point(0,-125*.48))
hourhand1 = Path(Point(0,0), Point(0,-125*.35))
hourhand1.rotate(5*30.0) # 5 o'clock * 30 degrees/hour
clock1.add(minutehand1)
clock1.add(hourhand1)
clock1.moveTo(150,75)
wall.add(clock1)

# Draw second clock, lightblue, diameter 200, time 8pm, at (350, 150):
clock2 = Layer()
clockface2 = Circle(200/2.0) # 200 is clockface diameter
clockface2.setFillColor('lightblue')
clock2.add(clockface2)
minutehand2 = Path(Point(0,0), Point(0,-200*.48))
hourhand2 = Path(Point(0,0), Point(0,-200*.35))
hourhand2.rotate(8*30.0) # 8 o'clock * 30 degrees/hour
clock2.add(minutehand2)
clock2.add(hourhand2)
clock2.moveTo(350,150)
wall.add(clock2)
```

Part (a) [12 points]

Capture the repeated pattern in the code above by creating a function, called **drawClock**, that can be used to draw clocks on the Canvas, such as those shown above. In part (b) on the next page, you will write three invocations of your **drawClock** function to draw the two clocks and then add a third in part (c).

Here, in part (a), you must define the **drawClock** function. Your **drawClock** function should take 6 parameters that provide the following information: the Canvas object to draw the clock on, the x-coordinate on the Canvas where the center of the clock should be placed, the y-coordinate on the Canvas where the center of the clock should be placed, the diameter of the clock, the color of the clock face, and the time the clock should display (an integer between 1 and 12).

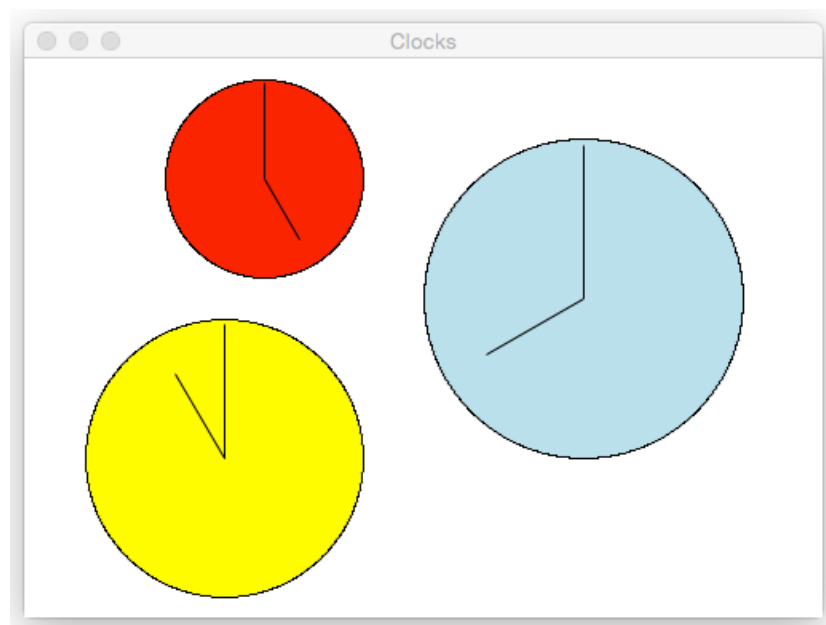
```
def drawClock(                                     ) :
```

Part (b) [6 points]

Write the two invocations of your **drawClock** function that will replace all of the code on page 2 except for the first two lines.

Part (c) [3 points]

Below, write one new invocation of your **drawClock** function that will additionally place a yellow clock of diameter 175 showing 11pm with its center at (125, 250) as shown in this image:



Write your invocation here:

Problem 2: Duplicate Removal [12 points]

Define a function named `removeDuplicates` that takes a list as its parameter and returns a new list consisting of all elements in the original list but without any duplicate elements. For example:

```
removeDuplicates([1, 2, 2, 3, 2, 3, 1])  
returns [1, 2, 3]
```

```
removeDuplicates(['john', 'george', 'paul', 'ringo', 'john',  
                 'paul', 'jones', 'rhys', 'price', 'jones' ])  
returns ['john', 'george', 'paul', 'ringo', 'jones', 'rhys', 'price']
```

```
removeDuplicates(range(0,10,2)+range(0,10,3))  
returns [0, 2, 4, 6, 8, 3, 9]
```

The order of elements in your returned list does not matter. For example, it is fine for your function to return `[2, 3, 1]` for the first example above.

Problem 3: Understanding Conditionals [16 points]

Fill in the table below to indicate the printed output for various values of the input variable **x**.

```
x = int(raw_input("Enter an integer x: "))
answer = ''
if x >= 60:
    answer = answer + 'A'
    if x < 75:
        answer = answer + 'B'
    elif x > 90:
        answer = answer + 'C'
    else:
        answer = answer + 'D'
elif x >= 50:
    answer = answer + 'E'
else:
    answer = answer + 'F'
if x < 35:
    answer = answer + 'G'
else:
    answer = answer + 'H'
    answer = answer + 'I'
print(answer)
```

x	printed output
25	
35	
45	
55	
65	
75	
85	
95	

Problem 4: Random Pairs [12 points]

In the Python `random` module, there is a function named `random.choice` that takes a list and returns a randomly selected element from that list. For example, here are some sample invocations:

```
myList = [8, 2, 7, 5]
random.choice(myList) returns 7
random.choice(myList) returns 8
random.choice(myList) returns 5
random.choice(myList) returns 5
random.choice(myList) returns 8
```

Write a function called `randomPair` that, given a list with at least two different elements, returns a pair (i.e., a tuple with two elements) of two distinct randomly selected elements from the given list.

`randomPair` should not modify the input list in any way. Here are some sample invocations of `randomPair`:

```
randomPair(myList) returns (5, 2)
randomPair(myList) returns (7, 8)
randomPair(myList) returns (2, 7)
randomPair(myList) returns (8, 5)
randomPair(myList) returns (7, 8)
```

Note that `randomPair(myList)` should never return (8,8) or any pair that contains two of the same element.

In your definition, employ this strategy: randomly choose the first element of the pair from the list, and then repeatedly randomly choose a second element from the list until that second element is different from the first element. You should not invoke any function other than `random.choice`

```
def randomPair(                                     ) :
```

Problem 5: Understanding Loops & Sequences [39 points]

Part (a) [12 points]

Define a function named **includesNumberBetween** that takes three parameters:

- (1) a low number
- (2) a high number
- (3) a list of numbers

It returns **True** if one of the numbers in the list is between the low number and the high number (inclusive). Otherwise it returns **False**.

```
includesNumberBetween(18, 40, [15, 7, 78, 63, 42]) returns False  
includesNumberBetween(40, 42, [15, 7, 78, 63, 42]) returns True  
includesNumberBetween(20, 100, [15, 7, 78, 63, 42]) returns True  
includesNumberBetween(65, 75, [15, 7, 78, 63, 42]) returns False  
includesNumberBetween(70, 80, [15, 7, 78, 63, 42]) returns True  
includesNumberBetween(15, 20, [15, 7, 78, 63, 42]) returns True
```

Define your function here:

Part (b) [15 points]

Consider the following functions `loopy1` and `test_loopy1`:

```
def loopy1 (nums):
    i = 0
    while i < (len(nums) - 2):
        nums[i] = nums[i] + nums.pop()
        i = i + 1

def test_loopy1():
    testList = [7, 4, 9, 2, 3]
    loopy1(testList)
    print(str(testList))
```

What is printed by the invocation `test_loopy1()`? To receive full credit, you **must** show your work, which can include iteration tables and memory diagrams.

Part (c) [12 points]

Below is a function `loopy2`:

```
def loopy2 (width, height, string):
    for y in range(height):
        line = ''
        for x in range(width):
            line = line + string[(y*width + x)%len(string)]
        print(line)
```

What is printed by the invocation `loopy2(5,3,'code')`?

This is the end of the exam.