CS111 EXAM 1
October 03, 2014

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):

❑Brian 9:50am ❑ Brian 11:10am ❑ Rhys 1:30pm

This exam has 6 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.**

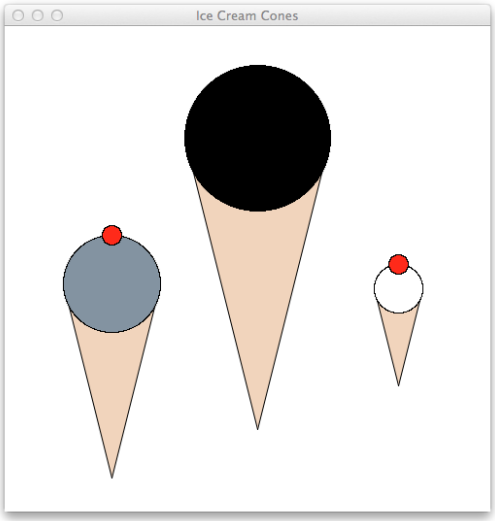
Please keep in mind the following tips:

- *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 **[18 pts]**	
Problem 2 **[12 pts]**	
Problem 3 **[20 pts]**	
Problem 4 **[18 pts]**	
Problem 5 **[20 pts]**	
Problem 6 **[12 pts]**	
Total [100 pts]	

Problem 1: Capture the Pattern [18 points]



I scream, you scream, we all scream for ice cream. Using **cs1graphics**, the canvas on the right can be produced with the code below.

```
1         from cs1graphics import *
2
3         paper = Canvas(500, 500,
4                 'white', 'Ice Cream Cones')
5
6         # Draw first ice cream cone dessert
7         dessert1 = Layer()
8         cone1 = Polygon(Point(-100/2,0), Point(100/2,0), Point(0,2*100))
9         cone1.setFillColor('peachpuff2')
10        dessert1.add(cone1)
11        scoop1 = Circle(100/2)
12        scoop1.setFillColor('darkgray')
13        dessert1.add(scoop1)
14        cherry1 = Circle(10, Point(0,-100/2)) # Cherries have radius 10
15        cherry1.setFillColor('red')    # Cherries are red
16        dessert1.add(cherry1) # Add cherry
17        dessert1.moveTo(110, 265)
18        paper.add(dessert1)
19
20        # Draw second ice cream cone dessert
21        dessert2 = Layer()
22        cone2 = Polygon(Point(-150/2,0), Point(150/2,0), Point(0,2*150))
23        cone2.setFillColor('peachpuff2')
24        dessert2.add(cone2)
25        scoop2 = Circle(150/2)
26        scoop2.setFillColor('black')
27        dessert2.add(scoop2)
28        dessert2.moveTo(260, 115)
29        paper.add(dessert2)
30
31        # Draw third ice cream cone dessert
32        dessert3 = Layer()
33        cone3 = Polygon(Point(-50/2,0), Point(50/2,0), Point(0,2*50))
34        cone3.setFillColor('peachpuff2')
35        dessert3.add(cone3)
36        scoop3 = Circle(50/2)
37        scoop3.setFillColor('white')
38        dessert3.add(scoop3)
39        cherry3 = Circle(10, Point(0,-50/2)) # Cherries have radius 10
40        cherry3.setFillColor('red')  # Cherries are red
41        dessert3.add(cherry3) # Add cherry
42        dessert3.moveTo(405, 270)
43        paper.add(dessert3)
```

Part (a) [12 points]

Capture the repeated pattern in the code above by creating a function, called **`drawIceCreamCone`**, that can be used to draw ice cream cones on the Canvas, such as those shown above. In part (b) on the next page, you will write three invocations of your **`drawIceCreamCone`** function to draw the three ice cream cones. Here, in part (a), you must define the **`drawIceCreamCone`** function. Your **`drawIceCreamCone`** function should take 6 parameters that provide the following information: the Canvas object to draw the cone on, the x-coordinate on the Canvas where the center of the cone should be placed, the y-coordinate on the Canvas where the center of the scoop (also the top of the Polygon cone) should be placed, the width of the cone which is also half the height of the cone, the color of the scoop of ice cream, and whether or not a cherry should be drawn on top of the scoop.

```
def drawIceCreamCone(                                                    ):
```

Part (b) [6 points]

Write the three invocations of your **`drawIceCreamCone`** function that will replace lines 7-18, lines 21-29, and lines 32-43 in the code above:

Problem 2: Lists [12 points]

Define a function named **`getUniqueListElements`** that takes two lists as parameters and returns a new list consisting of all elements that are in one of the two input lists but not both. For examples,

`getUniqueListElements([1,2,3,4,5,6],[2,4,6,8])` returns **`[1,3,5,8]`**
`getUniqueListElements(['hi','bye'],['bye','fly'])` returns **`['hi','fly']`**
`getUniqueListElements([1,2,3,4],[])` returns **`[1,2,3,4]`**
`getUniqueListElements(['cs111'],['cs111'])` returns **`[]`**

Problem 3: Lists and Functions [20 points]

Consider the following program.

```
def maxLength(list):
      max = 0
      for word in list:
            if len(word) > max:
                  max = len(word)
      return max

def categorize(list):
      newlist = []
      for i in range(1,maxLength(list)+1):
            sublist = []
            for word in list:
                  if  i == len(word):
                        sublist.append(word)
            newlist.append(sublist)
      return newlist

pronouns =['I','me','you','we','us','he','she','them','they','it']
print(str(categorize(pronouns)))
```

Part (a) [10 points]

Show the output of running the above program.

Part (b) [10 points]

Describe in English what the **`categorize`** function does.

Problem 4: Conditionals [18 points]

Many people think of leap years as years that are divisible by 4. While it is true that all years not divisible by 4 are not leap years, it is also true that not all years divisible by 4 are leap years. Years that are divisible by 100 are NOT leap years unless they are also divisible by 400, in which case they are leap years. For examples,

1999, 2001, 2002, 2003, 2005, 2006, 2007, 2009, 2010, 2011, 2013, 2014 are **not** leap years because they are not divisible by 4
1700, 1800, 1900, 2100, 2200, 2300, 2500 are **not** leap years because they are divisible by 100 and not divisible by 400
1600, 2000, 2400 are leap years because they are divisible by 400
1992, 1996, 2004, 2008, 2012, 2016, 2020 are leap years because they are divisible by 4 and not divisible by 100

Part (a) [12 points]

Define a function named **`isLeapYear`** that takes one parameter representing an integer year and returns **`True`** if the input year is a leap year and returns **`False`** otherwise.

Part (b) [6 points]

Define a function named **`getLeapYearsInRange`** that takes two integer parameters and returns a list of all leap years in the range between the first integer parameter, inclusive, and the second integer parameter, exclusive. For example,

`getLeapYearsInRange(1990,2020)` returns **`[1992,1996,2000,2004,2008,2012,2016]`**

Problem 5: Strings [20 points]

Part (a) [7 points]

Write a function named **`insert`** that takes two string parameters, **`s1`** and **`s2`**, and returns a new string similar to **`s2`** but with **`s1`** inserted into its middle. If **`s2`** has an odd length, then **`s1`** is inserted before the middle character in the returned string. For example,

`insert('X','cat')` returns **`'cXat'`**
`insert('X','oo')` returns **`'oXo'`**
`insert('ok','tiger')` returns **`'tiokger'`**
`insert('yuck','fart')` returns **`'fayuckrt'`**
`insert('yummy','chocolate')` returns **`'chocyummyolate'`**

Part (b) [7 points]

Write a function named **dropMiddle** that takes a string parameter and returns a new string similar to the input string but without the middle character(s). If the string parameter has an odd number of characters, then the returned string does not contain the middle character. If the string parameter has an even number of characters, then the returned string does not contain the two middle characters. For example,

dropMiddle('tiger') returns **'tier'**
dropMiddle('hand') returns **'hd'**
dropMiddle('axe') returns **'ae'**
dropMiddle('oh') returns **''**
dropMiddle('I') returns **''**

Part (c) [6 points]

Show what is printed by the following four lines of code:

```
print(insert('cool',dropMiddle('python')))

print(insert('hi',insert('bye',dropMiddle('ok'))))

print(insert(dropMiddle('ROAR'),'CS111'))

print(dropMiddle(insert('Y',dropMiddle('popsicle'))))
```

Problem 6: Parameters [12 points]

Consider the following program.

```
1          def leapRound(lf1, lf2, lf3):
2               print(str(lf1))
3               print(str(lf2))
4               print(str(lf3))
5               leap(lf1, lf3)
6               leap(lf2, lf1)
7
8          def leap(lf1, lf3):
9               lf1[0] = 1 + lf3[0]
10
11         leapFrog1 = [3,1]
12         leapFrog2 = [1,1]
13         leapFrog3 = [2,1]
14         leapRound(leapFrog2, leapFrog3, leapFrog1)
15
16         print(str(leapFrog1))
17         print(str(leapFrog2))
18         print(str(leapFrog3))
```

Part (a) [6 points]

When the program above is executed, show what will be printed by lines 2, 3, and 4.

Part (b) [6 points]

When the program above is executed, show what will be printed by lines 16, 17, and 18.