Name: _____

CSSE 120—INTRODUCTION TO SOFTWARE DEVELOPMENT

EXAM 1, SPRING 2009-2010 – **ANSWER KEY**

This exam contains two parts.

- The first part is to be done on paper without using your computer.
  - This part is closed-everything except you may use a "cheat sheet" that is one 8.5 by 11 sheet of paper (both sides, or two sheets each using only one side).
  - Complete this first part and turn it in before beginning the second part.
- The second part of the exam is to be done on your computer.
  - This part is open-everything except that you may not communicate with anyone except an exam proctor.
  - So, you can use Angel, the course web site, any projects on your computer, the entire World Wide Web, and search engines like Google, among other sources.
  - If you have questions on the on-the-computer part, you may ask an exam proctor. In particular, we don't expect you to be experts on understanding error messages yet, so if you get an error message, first try to understand it, but then ask for help as needed.
  - Obtain the on-the-computer part of this exam by checking out the **Exam1-201030** project from your SVN repository. It has 5 modules. Do the TODO's in each. Turn in your work by committing your project.

Time limit: You have a total of 2 hours to complete the entire exam, plus an hour of "bonus" time, so a total of up to 3 hours. Don't spend more than 30 minutes on the paper-and-pencil part.

Paper-and-pencil score:

Problem	Points	Score
P1	3	
P2	5	
P3	5	
P4	6	
P5	2	
P6	3	
Total	24	

Totals	Points	Score
paper	24	
computer	96	
Total	120	

On-the-computer score:

Problem	Points	Score	Explanation of points deducted
P1	10		
P2	10		
P3	10		
P4	10		
P5a	10		
P5b	10		
P5c	10		
P5d	10		
P5e	8		
P5f	8		
Total	96		

Your total as a percent (of 100): _____

Name: _____

PROBLEMS

1. (3 points) What is the value of each of the following expressions?

```
11 % 3
```

Value: **2**

```
11 / 3
```

Value: **3**

```
float(11 / 3)
```

Value: **3.0**

2. (5 points: 1, 1 and 3, respectively) What is the output of each of the following code fragments?

```
x = 3
x *= 2
print x + x
```

Output: **12**

```
def fun(x):
    a = 2
    return a + x

a = 8
b = fun(a)
print a, b
```

Output: **8, 10**

```
def g(a, b):
    print a, "of" , b

def f(n):
    for i in range(1, n + 1):
        g(i, n)
    return n ** 2, n ** 3

x, y = f(3)
print x + y
```

1 point

1 point

Output: **1 of 3**
2 of 3
3 of 3
36

1 point

Name: _____

3. (5 points) What gets printed by the following expressions:

```
range(2, 11, 3)
```
[2, 5, 8]

On these problems, OK if they have the right idea. E.g., OK if omitted []s or outermost quotes, and any form of quotes is OK.

```
""" 'hello' """
```
" 'hello' "

```
"one" + "two"
```
"onetwo"

```
x = [10, 20, 30, 40, 50]
print x[2]
```
30

```
y = "forget me not"
print y[2]
```
'r'

---

4. (6 points) For each of the following lists, write a Python `range` expression that produces the list:

`[1, 2, 3, 4, 5]` **range(1, 6)**

On these problems, half credit for any single error, no credit if two or more errors.

`[4, 6, 8, 10, 12]` **range(4, 13, 2)**

`[5, 2, -1, -4, -7]` **range(5, -8, -3)**

Name: _____

5. (2 points) What gets printed by the following expressions:

```
quotesList = ["Immature poets imitate; mature poets steal.",
              "by T.S. Eliot"]

print quotesList[1]
```

"by T.S. Eliot"

On these problems, OK if quotes are omitted.

```
print quotesList[1][1]
```

"y"

---

6. (3 points) What gets printed by the following code:

```
circleA = Circle(Point(25, 25), 10)
circleB = circleA
circleA.move(15, 0)
print circleA.getCenter().getX()
print circleB.getCenter().getX()
```

1 point

Output: 40
40

2 points

When you have completed this part of the exam, turn it in to the exam proctor so you can use your computer for the next (on-the-computer) part of the exam.

Begin the on-the-computer part by checking out the project called
Exam1-201030
in your SVN repository. Do the five problems therein, each of which has TODO's that specify what to do. Commit your work when you are done.

Recall the instructions on the computer part: Open everything EXCEPT that you may NOT communicate with anyone except the exam proctors.

Name: _____

```
'''
Exam 1, problem 1.
This problem lets you demonstrate your understanding of:
  -- the input/compute/output pattern
  -- using functions from a library
  -- conditionals.
Authors: David Mutchler and DONE: 1. Answer Key by David Mutchler.
Created on Apr 1, 2010.
'''

import math

# DONE: 2. In the space immediately below this comment,
# write a function called ** sine_or_cosine ** that takes NO parameters.
# It should prompt for and input a number from the console.
# The function prints either the cosine of the number
# or the sine of the number, whichever is larger.
def sine_or_cosine():
    x = input("Enter a number: ")
    if math.cos(x) > math.sin(x):
        print math.cos(x)
    else:
        print math.sin(x)

def main():
    ''' Tests the sine_or_cosine function '''
    # DONE: 3. In the space immediately below this comment,
    # write a statement that calls your sine_or_cosine function
    # (so that you can test it).
    sine_or_cosine()

main()
```

1 point

2 points

4 points

2 points

1 point

2 points

Name: _____

```
'''
Exam 1, problem 2.
This problem lets you demonstrate your understanding of:
  -- using functions from a library
  -- writing and calling functions
  -- string operations
Authors: David Mutchler and DONE: 1. Answer Key by David Mutchler.
Created on Apr 1, 2010.
'''

# DONE: 2. In the space immediately below this comment,
# write a function called ** string_lengths ** that
# takes THREE parameters, which should be strings.
# The function returns the sum of the lengths of the three strings.
# For example, string_lengths("one", "two", "three")
# should return 11.
def string_lengths(s1, s2, s3):
    return len(s1) + len(s2) + len(s3)

def main():
    ''' Tests the string_lengths function '''
    # DONE: 3. In the space immediately below this comment,
    # write code that tests your string_lengths function.
    print string_lengths("one", "two", "three")
    print string_lengths("", "", "")

main()
```

3 points

3 points

3 points

3 points

Name: _____

```
'''
Exam 1, problem 3.
This problem lets you demonstrate your understanding of:
  -- writing and calling functions
  -- the accumulator pattern
  -- using functions from a library
Authors: David Mutchler and DONE: 1. Answer Key by David Mutchler.
Created on Apr 1, 2010.
'''

import math

def sum_of_square_roots(n):
    '''
    Returns:
    the square root of 1 + the square root of 2
    + the square root of 3 + ... + the square root of n
    '''
    # DONE: 2. Implement this function.
    sum = 0
    for k in range(1, n + 1):
        sum = sum + math.sqrt(k)
    return sum

def main():
    ''' Tests the sum_of_square_roots function '''

    print "sum_of_square_roots(%d) returns %s, which should be about %s.  Is it?" \
    % (100, sum_of_square_roots(100), 671.462947103)

    print "sum_of_square_roots(%d) returns %s, which should be %s.  Is it?" \
    % (0, sum_of_square_roots(0), 0)

    print "sum_of_square_roots(%d) returns %s, which should be %s.  Is it?" \
    % (1, sum_of_square_roots(1), 1.0)

    # DONE: 3. (Optional) Add additional tests if you feel they are needed.

main()
```

1 point

2 points

3 points

3 points

3 points

Name: _____

```
'''
Exam 1, problem 4.
This problem lets you demonstrate your understanding of:
  -- writing and calling functions
  -- returning multiple values from a function
  -- string indexing
  -- the accumulator pattern
Authors: David Mutchler and DONE: 1. Answer Key by David Mutchler.
Created on Apr 1, 2010.
'''

def odds_and_evens(s):
    '''
    Takes a string s, and returns TWO values:
    the string formed from the odd-numbered characters of s, and
    the string formed from the even-numbered characters of s.
    For example, odds_and_evens("This is a test") should return
    "nsi  et" and "Ti sats".
    '''
    # DONE: 2. Implement this function.
    evens = ""
    for k in range(0, len(s), 2):
        evens = evens + s[k]

    odds = ""
    for k in range(1, len(s), 2):
        odds = odds + s[k]

    return odds, evens

def main():
    ''' Tests the odds_and_evens function '''
    odds, evens = odds_and_evens("This is a test")
    print odds, "-- this should be: hsi  et"
    print evens, "-- this should be: Ti sats"

    odds, evens = odds_and_evens("01234567890")
    print odds, "-- this should be: 13579"
    print evens, "-- this should be: 024680"

    # DONE: 3. (Optional) Add additional tests if you feel they are needed.

main()
```

1 point

2 point

2 points

1 point

2 point

2 points

2 points

Name: _____

```
'''
Exam 1, problem 5.
This problem lets you demonstrate your understanding of:
  -- constructing objects
  -- using objects' methods
  -- using zellegraphics to draw shapes
  -- animation in zellegraphics
Authors: David Mutchler and DONE: 0. Answer Key by David Mutchler.
Created on Apr 1, 2010.
'''

from zellegraphics import * #@UnusedWildImport
import math

def many_shapes(window):
    '''
    This function draws various objects, as described below.
    All drawing is on the given window.
    '''
    # DONE: 1. Implement (and test) Stage 1:
    # A blue circle appears.  It has a red outline,
    # is centered at (50, 20) and has radius 10.
    #
    # For ALL these stages, if you have questions about WHAT a stage is
    # asking you to do, don't hesitate to ask your instructor to demo the stage.
    red_circle = Circle(Point(50, 20), 10)
    red_circle.setFill('blue')
    red_circle.setOutline('red')
    red_circle.draw(window)

    # DONE: 2. Implement (and test) Stage 2:
    # When the user clicks the mouse, a single black circle appears,
    # centered on the mouse click, with radius 10.
    mouse_point1 = window.getMouse()
    red_circle = Circle(mouse_point1, 10)
    red_circle.setFill('black')
    red_circle.draw(window)

    # DONE: 3. Implement (and test) Stage 3:
    # When the user clicks the mouse a second time,
    # a line appears, from the point of the first mouse click
    # to the point of the second mouse click.  The line
    # should have width 5 (so it is a thick line).
    mouse_point2 = window.getMouse()
    line = Line(mouse_point1, mouse_point2)
    line.setWidth(5)
    line.draw(window)
```

Each TODO (other than the student's name) is 8 points:

- 4 points if there is a single error
- 0 points if there is more than one error, or if the answer is just way off

Judge correctness by running the program. It is OK if style is poor or they solve the problem in an unusual way (but consider making a comment to them on the cover page). For any points deducted, mark the reason on the cover page.

Name: _____

```
    # DONE: 4. Implement (and test) Stage 4:
    # After the user has clicked the mouse twice
    # (and hence the line between the clicks appears),
    # a circle appears.  Its center is the center of the line
    # from the previous stage, and its diameter is the length
    # of the line from the previous stage. Its outline is red
    # and it is NOT filled (so you can see the line inside it).
    #
    # For full credit, use the appropriate method from the Line class
    # to obtain the center of the line, and use the distance function
    # defined below to get the length of the line.
    center_of_line = line.getCenter()
    length_of_line = distance(line.getP1(), line.getP2())
    red_circle = Circle(center_of_line, length_of_line / 2)
    red_circle.setOutline('red')
    red_circle.draw(window)

    # DONE: 5. Implement (and test) Stage 5:
    # Now 10 more circles appear, all of them green.
    # They all have the same radius as the circle in the previous stage.
    # They form a line of circles, starting with the circle from the
    # previous stage and moving off to the right, each circle just
    # touching its neighbor circles.  It is OK if the circles go beyond
    # the dimensions of the window.
    x = red_circle.getCenter().getX()
    y = red_circle.getCenter().getY()
    radius = red_circle.getRadius()

    for k in range(10): #@UnusedVariable
        x += radius * 2
        circle = Circle(Point(x, y), radius)
        circle.setFill('green')
        circle.draw(window)
```

Name: _____

```
82
83      # DONE: 6. Implement (and test) Stage 6:
84      # Finally, a yellow circle appears, directly below the leftmost green circle,
85      # with the same radius as the green circles.
86      # It "drops" smoothly down until it is at the bottom of the window,
87      # so that the entire circle is displayed on the window but dropping it down
88      # one more step would make part of the circle go off of the window.
89      # The best solution will work correctly no matter what dimensions the window has.
90      radius = red_circle.getRadius()
91      x = red_circle.getCenter().getX() + radius * 2
92      y = red_circle.getCenter().getY() + 4 * radius
93      circle = Circle(Point(x, y), radius)
94      circle.setFill('yellow')
95
96      n_circles = round(window.getHeight() - y - radius)
97      for k in range(int(n_circles)): #@UnusedVariable
98          circle.draw(window)
99          time.sleep(0.01)
100         circle.undraw()
101         circle.move(0, 1)
102     circle.draw(window)
103
104 def distance(point1, point2):
105     ''' Returns the distance between the two points, as an integer. '''
106     squareX = (point1.getX() - point2.getX()) * (point1.getX() - point2.getX())
107     squareY = (point1.getY() - point2.getY()) * (point1.getY() - point2.getY())
108     return round(math.sqrt(squareX + squareY))
109
110 def main():
111     ''' Tests the many_circles function '''
112     problem5_window = GraphWin("Problem 5", 800, 600)
113     many_shapes(problem5_window)
114     problem5_window.getMouse()    # Keep the window displayed until you click on it
```