Name: ____Solution Key_____

CSSE 120—INTRODUCTION TO SOFTWARE DEVELOPMENT

EXAM 2, SPRING 2009-2010

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 **Exam2-201030** 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	5	
P2	5	
P3	5	
P4	3	
P5	5	
P6	7	
Total	30	

Totals	Points	Score
paper	30	
computer	70	
Total	100	

On-the-computer score:

Problem	Points	Score	Explanation of points deducted
P1	19		
P2	19		
P3	19		
P4	13		
Total	70		

1. (5 points) What gets printed by the following code?

```
def silly(x):
    print "start silly"
    print x
    goofy(x - 1)
    funny(x + 2)
    print "end silly"

def funny(y):
    print "start funny"
    print y
    goofy(y * 2)
    print "end funny"

def goofy(x):
    print "start goofy"
    print x
    print "end goofy"

silly(3)
```

Output:

-1 for each error, but not less than zero total.

```
start silly
3
start goofy
2
end goofy
start funny
5
start goofy
10
end goofy
end funny
end silly
```

2. (5 points) What gets printed by the following code?

```
a = 8
b = 64
while a < b:
    print a, b
    a = a + 1
    b = b / 4
```

Output:

-2 for each error, but not less than zero total.

```
8 64
9 16
```

Name: ___Solution Key____

3. (5 points) What gets printed by the following code?

```
def foo(x, y, z):
    x = x + z[1]
    y = [1, 2, 3]
    z[0] = z[1]
    z[1] = z[2]
    print x, y, z

a = 4
b = [5, 10, 15]
c = [8, 30, 60]
foo(a, b, c)
print a, b, c
```

Output:

-1 for each error, but not less than zero total.

```
34 [1, 2, 3] [30, 60, 60]
4 [5, 10, 15] [30, 60, 60]
```

4. (3 points) What gets printed by the following code?

```
circleA = Circle(Point(25, 25), 10)
circleB = Circle(Point(50, 50), 20)
circleC = Circle(Point(75, 75), 30)

circleC = circleB
circleB = circleA

circleA.move(100, 0)
circleB.move(200, 0)

print circleA.getCenter().getX()
print circleB.getCenter().getX()
print circleC.getCenter().getX()
```

Output:

-1 for each error

```
325
325
50
```

5. (5 points) Explain what a *sentinel* value is and give a concrete example of its use.

A sentinel is a special value that signals the end of a loop. It is often used in a loop that gets input from the user until the user types the special sentinel value.

6. (7 points) Write a function called *between* that takes a list of numbers and two integers, and returns a list of all numbers in the given list that are between the two integers (including the endpoints). You may assume that the first of the two integers is less than or equal to the second of the two integers.

Here is an example:

```
between([17, 22, 8, 0, 13, 18, 6, 12, 21, 25], 13, 22)
```

returns `[17, 22, 13, 18, 21]`

```
def between(numbers, x, y):
    result = []
    for number in numbers:
        if number >= x and number <= y:
            result.append(number)
    return result
```

-2 for each error, but not less than zero total

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

Exam2-201030

in your SVN repository. Do the 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: ___Solution Key____

```
'''
Exam 2, problem 1.
Authors: David Mutchler and TODO: 1. Solution by David Mutchler.
Created on Apr 1, 2010.
'''

# TODO: 2. Write a function called rows_and_columns
# that has two parameters, r and c.  It prints r rows
# and c columns that looks like this example in which
# r is 6 and c is 5. (But your function has to work for
# any reasonable r and c, not just 6 and 5.)
#
# 1 2 3 4 5
# 11 12 13 14 15
# 21 22 23 24 25
# 31 32 33 34 35
# 41 42 43 44 45
# 51 52 53 54 55
#
# Your numbers can be against the left edge of the console -
# they don't have to have a space before the first number,
# as the above example might suggest.  Also, you can earn
# 2/3 credit if your numbers go in this perhaps-easier pattern:
# 1 2 3 4 5
# 2 3 4 5 6
# 3 4 5 6 7
# etc
def rows_and_columns(r, c):
    for row in range(r):
        for column in range(c):
            print 10 * row + column + 1,
            #  or:  print str(row) + str(column + 1),
            # with a special case for row 0
        print

def main():
    ''' Tests the rows_and_columns function. '''
    # TODO: 3. Write code here that tests your rows_and_columns function.
    rows_and_columns(6, 5)
    rows_and_columns(4, 12)

if __name__ == '__main__':
    main()
```

```
'''
Exam 2, problem 2.
Authors: David Mutchler and TODO: 1. Solution by David Mutchler.
Created on Apr 1, 2010.
'''

def positives_and_negatives():
    '''
    Prompts for and inputs numbers from the user, one number per line,
    until the user enters a 0.
    Returns two lists: the list of positive numbers entered by the user,
    and the list of negative numbers entered by the user.
    You may assume that the user enters only numbers (no malformed data).

    You may earn 1/2 credit if you return just the positives list.
    (In that case, you will need to modify main as indicated below - get help
    '''
    # TODO 2: Implement this function.
    positives = []
    negatives = []
    while (True):
        number = input("Enter a number: ")
        if number == 0:
            break;
        if number < 0:
            negatives.append(number)
        else:
            positives.append(number)
    return positives, negatives

def main():
    ''' Tests the positives_and_negatives function. '''
    positives, negatives = positives_and_negatives()
    print "Positives:", positives
    print "Negatives:", negatives

    # If you do the just-positives version of the problem (for 1/2 credit),
    # comment out the above lines and uncomment out the two lines below.
    # positives = positives_and_negatives()
    # print "Positives:", positives

if __name__ == '__main__':
    main()
```

```
'''
Exam 2, problem 3.
Authors: David Mutchler and TODO: 1. Solution by David Mutchler.
Created on Apr 1, 2010.
'''

def make_student_dictionaries():
    '''
    Returns a list of students.  Each student is a dictionary
    with keys for the student's first name, age,
    and possibly their car (if the student owns one).
    '''
    students = []
    d = {'first_name' : 'bob', 'last_name' : 'martin', 'age' : 19, 'car' : '1985 mustang'}
    students.append(d)

    d = {'first_name' : 'suzy', 'last_name' : 'martin', 'age' : 21, 'car' : '2002 civic'}
    students.append(d)

    d = {'first_name' : 'suzy', 'last_name' : 'fisher', 'age' : 17}
    students.append(d)

    d = {'first_name' : 'susie', 'last_name' : 'king', 'age' : 19, 'car' : '2000 sable'}
    students.append(d)

    d = {'first_name' : 'suzy', 'middle_name' : 'unknown'}
    students.append(d)

    return students

def return_matching(students, name_to_look_for):
    '''
    Students is a list of dictionaries, with each dictionary
    having a 'first_name' key.  This function returns a list of all
    the students whose name equals the name_to_look_for.
    Do NOT assume that students is the particular list generated above;
    it can be any list of dictionaries, as long as each dictionary
    has a 'first_name' key.
    '''
    # TODO 2: Implement this function.
    result = []
    for student in students:
        if student['first_name'] == name_to_look_for:
            result.append(student)
    return result
```

Name: ___Solution Key____

```
'''
Exam 2, problem 4.
Authors: David Mutchler and TODO: 1. Solution by David Mutchler.
Created on Apr 1, 2010.
'''

def biggest_run_sum(list_of_numbers):
    '''
    This function is given a list of numbers.
    A "run" in this list is a sequence of entries all of which are nonzero;
    that is, a zero ends a run and a nonzero entry begins a run.

    For example, [8, 4, 12, 9, 0, 170, 4, 0, 5, 5, 5, 0] has 3 runs,
    while [10, 20, 0] has just one run, and [10, 20, 0, 5, 6, 3, 0] has two runs.
    Ask me for more examples if you are not clear what a "run" is.

    For each run, add up the numbers in the run to get the run's sum.
    Return the largest of these sums.

    For example, for the list above that has 3 runs, the run's have sums 33, 174, and 15,
    so the function would return 174 on that list.  See main below for more examples.

    You may assume that the list is non-empty, that it does not begin with a zero,
    and that it ends with a zero.
    '''
    # TODO 2: Implement this function.
    sum = 0
    max = 0
    for number in list_of_numbers:
        if number == 0:
            if sum > max:
                max = sum
            sum = 0
        else:
            sum = sum + number
    return max

def main():
    ''' Tests the biggest_run_sum function. '''
    print biggest_run_sum([8, 4, 12, 9, 0, 170, 4, 0, 5, 5, 5, 0])
    # The above should print: 174

    print biggest_run_sum([10, 20, 0, 8, 4, 12, 9, 0, 20, 4, 0, 5, 5, 5, 5, 5, 5, 0])
    # The above should print: 33

    print biggest_run_sum([10, 20, 0])
    # The above should print: 30
```