Exam 1 – Paper and Pencil part

Have you completed and committed programming exercises *m1* through *m3* from Session 7?

If not, DO NOT BEGIN THIS EXAM!

Instead, see your instructor to find out where you should go to complete the Session 7 problems.

Name: ***SOLUTION and guide to partial credit***

Section: 1 2 3 4

1 = Mutchler, 1st-2nd periods. **2 =** Mutchler, 3rd-4th periods.
3 = Anderson, 7th-8th periods. **4 =** Anderson, 9th-10th periods.

Honesty Pledge:

At the beginning of the exam, you will receive an ***Honesty Pledge*** that is exactly the same as the one which you should have read before the exam. Re-read the Honesty Pledge at the beginning of the exam. ***Sign and turn in that pledge when you finish this exam.***

Two parts:

Part 1: ***Paper-and-Pencil.***

For this part, the ONLY external resource you may use is a single 8½ by 11-inch sheet of paper, with whatever you want on it, typed or handwritten or a combination of the two. You may use BOTH sides of the sheet. You must have prepared the sheet *before* beginning the exam.

Part 2: ***On-the-computer.***

For this part, the only external resources allowed are:

- Any written material you choose to bring to the exam: books, handouts, notes, etc.
- Your computer and anything on it.
- Your own SVN repository
- Anything directly reachable from the CSSE 120 web site.

You may *not* use any electronic device other than your computer.

You may *not* use any search engine (like Google). (Exception: If a search engine is embedded into a site directly reachable from the CSSE 120 web site, and is restricted to that site, then you may use that limited search engine.)

Communication:

For both parts of the exam, ***you must not communicate with anyone*** except your instructor and his delegates, if any.

Time limit:

You have ***two hours*** to complete the entire exam – its *paper part* and its *computer part*. You will receive both parts at the beginning of the exam. You must complete the paper part (using only your prepared 1-page-front-and-back sheet) and turn it in before you begin work on the computer part.

Problem	Points Possible	Points Earned
1	5	
2	3	
3	4	
4	4	
5	4	
6	5	
Total	25	

1. (5 points) Consider the code snippet to the right. It is a contrived example with poor style, but it will run without errors. What does it print when *main* runs?

```
def main():
    first(9)
    second(6)
    third(3)

def first(x):
    print('first')
    if x < 8:
        print(x)
    else:
        print(x * x)

def second(x):
    print('second')
    print(4 * x)

def third(x):
    print('third')
    first(2 * x)
    second(10 * x)
```

Output:

```
First
81
Second
24
Third
First
6
Second
120
```

Notes: main calls first(6),
In first, x is 9.
Bigger than 8, so 81 is printed.
main calls second(6).
In second, x is 6,
so 24 is preinted.
main calls third(3)
in third, x is 3.
third calls first(6)
in first, x is 6, so 6 is printed
third calls second(30)
in second, x is 30, so print 120

Partial credit:

- **The student earns 1 point for each of the 4 numbers (regardless of order)**
- **The student earns 1 more point for the words (all or nothing)**
- **Subtract 1 additional point if one number is in the wrong order. Subtract 2 points if more than one number is in the wrong order. (E.g., the student earns 3 of 5 points if the numbers and words are right but are badly screwed up regarding the order in which they appear.)**
- **Ignore "obvious" arithmetic or clerical errors (e.g., full credit if they write 150 on the last one but all else is OK)**

2. (3 points) Consider the code snippet to the right. It is a contrived example with poor style, but it will run without errors. What does it print when *main* runs?

   Write your answer in the box to the right of the code.

```
def main():
    x = 7
    y = foo(x)
    print(x)
    print(y)

def foo(x):
    x = 10
    print(x)
    return 3 * x
```

Output:

```
10
7
30
```

Notes: main calls foo(7),
foo's x is 6, changed to 10
print 10
Return 30
Back in main, returned 30 becomes the value of y.
main's x is still 7,
print 7
print 30

Partial credit:

- **The student earns 1 point for each of the 3 numbers (regardless of order)**
- **Subtract 1 point if one or more of the numbers are in the wrong order.**
- **Ignore "obvious" arithmetic or clerical errors**

3. (4 points) Consider the code snippet to the right. What does it print when it runs?

   Write your answer in the box to the right of the code.

```
t = 3
for k in range(2, 8, 2):
    t = t + k
    print(t)

print(t)
```

Output:

5
9
15
15

Notes:
t starts out as 3.
First time through loop, k is 2, so t becomes 5. Print 5.
Second time through loop, k is 4, so t becomes 9. Print 9.
Third time through loop, k is 6, so t becomes 15. Print 15.
After loop, print 15 again.

Partial credit:

- **The student earns 1 point for the first number (5)**
- **The student earns 1 more point if the second number is 4 bigger than the first number.**
- **The student earns 1 more point if the third number is 6 bigger than the second number.**
- **The student earns 1 more point if the fourth number is the same as the third number.**
- **Subtract 1 point if 1 too many numbers. Subtract 2+ points if more than 1 too many numbers.**
- **Ignore "obvious" arithmetic or clerical errors**

4. (4 points) For each of the following Boolean expressions, indicate whether it evaluates to ***True*** or ***False*** (circle your choice):

True	(*False*)	`not (4 < 8)`
True	(*False*)	`(not (4 < 8)) and (5 < 7)`
(*True*)	*False*	`(not (4 < 8)) or (5 < 7)`
True	(*False*)	`(6 <= 6) and (3 == 2)`

5. (4 points – 2, 1 and 1, respectively)

   a. Write two Python constants – one an integer (**`int`**) and one a floating point number (**`float`**) – that clearly shows the difference between the **`int`** and **`float`** types.

   4 3.1 Any two numbers – one with a decimal point and one without – are correct. If anyone says *math.pi* I suppose that that is correct to (for the float).
   Partial credit: Part (a) is 2 points – mostly all or nothing (ask if you think otherwise).

   b. There is a limit to the number of digits a Python **`int`** can have. *True* (*False*) (circle your choice)

   c. There is a limit to the number of significant digits a Python **`float`** can have. (*True*) *False* (circle your choice)

6. (5 points) Consider a function whose name is ***print_string*** that takes two arguments as in this example:

```
print_string('Robots rule!', 4)
```

The function should print the given string the given number of times. So, the above function call should produce this output:

```
Robots rule!
Robots rule!
Robots rule!
Robots rule!
```

Write (in the space below) a complete implementation, ***including the header (def) line***, of the above ***print_string*** function.

```
def print_string(s, n):
    for k in range(n):
        print(s)
```

Partial credit:

- **Full credit regardless of what variable names they choose (even not-so-great-ones like the ones that the above solution has). *[CLAUDE, DO YOU THINK OTHERWISE?]***
- **Full credit if indentation is sloppy but appears to be mostly right.**
- **For the 5 points, the student earns:**
  - **1 point for `def print_string` [full credit if `print_string` is without the underscore or *slightly* misspelled]**
  - **2 point for the parameters in the header line (order matters!) AND using them correctly in the FOR and PRINT statements.**
    - **The student earns 1 of these 2 points if there is a single small error, like putting the right header line and using s correctly but not n. The student earns 0 of these 2 points for any more severe error(s) regarding the parameters.**
  - **1 point for the FOR statement [but full credit for any errors that we would call "semi-colon errors" in Java or C]**
  - **1 point for the PRINT statement [but full credit for any errors that we would call "semi-colon errors" in Java or C]**

CSSE 120 201410 Exam 1 Computer part Problem 1

```
    f = float(input('Enter a number: '))
    s = input('Enter a string: ')
    print(math.cos(f))
    print(s, s)
```

Problem 2

```
def test_sale_price():

    amount = 20.00
    price = sale_price(amount)
    print('purchase amount is', amount,
          '  sale price: expected = 18.00, calculated =', price)

    amount = 50.00
    price = sale_price(amount)
    print('purchase amount is', amount,
          '  sale price: expected = 40.00, calculated =', price)

    amount = 130.00
    price = sale_price(amount)
    print('purchase amount is', amount,
          ' sale price: expected = 91.00, calculated =', price)

    amount = 300.00
    price = sale_price(amount)
    print('purchase amount is', amount,
          ' sale price: expected = 180.00, calculated =', price)

def sale_price(purchase_amount):
    if purchase_amount < 50:
        sale = purchase_amount * 0.9
    elif purchase_amount < 100:
        sale = purchase_amount * 0.8
    elif purchase_amount < 200:
        sale = purchase_amount * 0.7
    else:
        sale = purchase_amount * 0.6

    return sale
```

Problem 3

```
def problem3a(m, n):

    total = 0
    for k in range(m, n + 1):
        total = total + math.factorial(k)
    return total


def problem3b(m):
    count = 0
    for k in range(m, (m ** 3) + 1):
        if largest_factor(k) % 2 == 1:
            count = count + 1
    return count
```

Problem 4

```
def problem4a(window, center_square):
    center_square.draw(window)
    p1 = center_square.getP1()
    p2 = center_square.getP2()
    width = p2.getX() - p1.getX()

    above_square = zg.Rectangle(zg.Point(p1.x, p1.y - width),
                                zg.Point(p2.x, p2.y - width))
    above_square.draw(window)
    below_square = zg.Rectangle(zg.Point(p1.x, p1.y + width),
                                zg.Point(p2.x, p2.y + width))
    below_square.draw(window)
    left_square = zg.Rectangle(zg.Point(p1.x - width, p1.y),
                               zg.Point(p2.x - width, p2.y))
    left_square.draw(window)
    right_square = zg.Rectangle(zg.Point(p1.x + width, p1.y),
                                zg.Point(p2.x + width, p2.y))
    right_square.draw(window)


def problem4b(dot_count):
    win = zg.GraphWin('Click anywhere to place a dot', 300, 300)
    radius = 5
    for k in range(dot_count):  # @UnusedVariable
        clickPoint = win.getMouse()
        cir = zg.Circle(clickPoint, radius)
        cir.setFill('darkblue')
        cir.draw(win)
    win.closeOnMouseClick()
```