Name: ____Sample_Solution_____

Email address (UW NetID): _____

CSE 160 Spring 2015: Midterm Exam

(closed book, closed notes, no calculators)

Instructions: This exam is closed book, closed notes. You have 50 minutes to complete it. It contains 12 questions and 11 pages (including this one), totaling 72 points. Before you start, please check your copy to make sure it is complete. Turn in all pages, together, when you are finished. Please write neatly; we cannot give credit for what we cannot read.

Good Luck! Total: 72 points. Time: 50 minutes.

Problem	Score
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
Total	

1) [4 pts] What is the **result** and **type** of the following expressions (if it is an Error indicate that):

	Result	Type	Error? (yes/no)
`20/10`	**2**	**integer**	**No**
`10/20`	**0**	**integer**	**No**
`1.0/4`	**0.25**	**float**	**No**
`45 < 90`	**True**	**bool**	**No**

2) [3 pts] What is the last element generated by these `range` () calls:

	Last Element Generated
`range(5, 25, 5)`	**20**
`range(14, 3, -2)`	**4**
`range(13)`	**12**

3) [3 pts] How many times will the **`print`** statement be executed in the code below?

```
for x in [2, 4, 6, 8]:
    for y in range(x):
        print str(x) + "!"
```

MY ANSWER:

20

4) [3 pts] Rewrite the following nested if statements as a single if statement (may contain elif and else)?

```
if val > 120:
    print "Green!"
else:
    if val > 80:
        print "Blue!"
    else:
        if val < 32:
            print "Red!"
        else:
            print "Orange!"
```

MY ANSWER:

Several possible correct answers, here are two:

```
if val > 120:
    print "Green!"
elif val > 80:
    print "Blue!"
elif val < 32:
    print "Red!"
else:
    print "Orange!"
```

```
if val > 120:
    print "Green!"
elif val > 80:
    print "Blue!"
elif val >= 32:
    print "Orange!"
else:
    print "Red!"
```

5) [6 pts] What output is produced after running the following piece of code?
(Hint: You may find it useful to draw out the environment frames.)

```
a = 12
p = 6

def foo(a):
    a = cat(a + 1)
    print "In foo", a, "and", p

def bar(i):
    i = i + 2
    print "In bar", i

def cat(a):
    a = a + 10
    bar(a)
    print "In cat", a
    return a


print "foo(p)=", foo(p)
print "a=", a
print "p=", p
```

MY ANSWER:

foo(p)= In bar 19
In cat 17
In foo 17 and 6
None
a= 12
p= 6

6) [4 pts] Given the following dictionary, write what each expression evaluates to. If an error is thrown, write "**Error**".

```
my_dict = {"one":1, 2:2, 'string':'with', 'withhold':-100, 4:16 }
```

a) `my_dict['1']` — **KeyError: '1'**

b) `my_dict[6-2]` — **16**

c) `my_dict[-100]` — **KeyError: -100**

d) `my_dict[my_dict['string']+'hold']` — **-100**

7) [4 pts] What is the output of the following code? If the code has an error write "**Error**".

```
a = {'do', 'ray', 'me'}
b = {'fa', 'do'}
```

Note: the elements in these sets can be in any order.

a) `print a - b` — **set(['me', 'ray'])**

b) `print a & b` — **set(['do'])**

c) `print a | b` — **set(['me', 'do', 'fa', 'ray'])**

d) `print set('pizza')` — **set(['i', 'p', 'z', 'a'])**

8) [4 pts] What output is produced after running the following piece of code?

```
from operator import itemgetter

def some_key(info):
    return itemgetter(2,0)(info)

favorites = [ ('Ann', 'Yellow', 4), ('Terry', 'Yellow', 5),
              ('George', 'Green', 3), ('Gene', 'Blue', 3),
              ('Jane', 'Red', 5) ]

print sorted(favorites, key = some_key)
```

MY ANSWER:

[('Gene', 'Blue', 3), ('George', 'Green', 3),
('Ann', 'Yellow', 4), ('Jane', 'Red', 5),
('Terry', 'Yellow', 5)]

9) [3 pts] After this code executes, what is printed?

```
list_A = [17, 3]
list_A.append(5)
list_B = list_A
list_B.append(21)
list_C = list(list_A)
list_A.append(8)
list_C.append(4)

print list_A
print list_B
print list_C
```

MY ANSWER:

[17, 3, 5, 21, 8]

[17, 3, 5, 21, 8]

[17, 3, 5, 21, 4]

10) [8 pts] a) **Draw** the entire environment, including all active environment frames and all user-defined variables, **at the moment that the PLUS OPERATION IS performed**. Feel free to draw out the entire environment, but be sure to CLEARLY indicate what will exist at the moment the **PLUS** operation is performed (e.g. cross out frames that no longer exist).

b) When finished executing, **what is printed out by this code**?

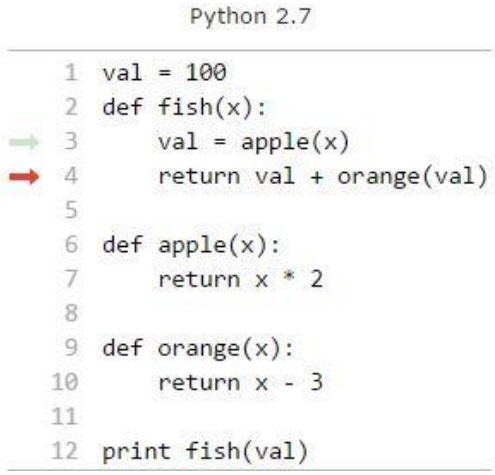
MY ANSWER: **397**

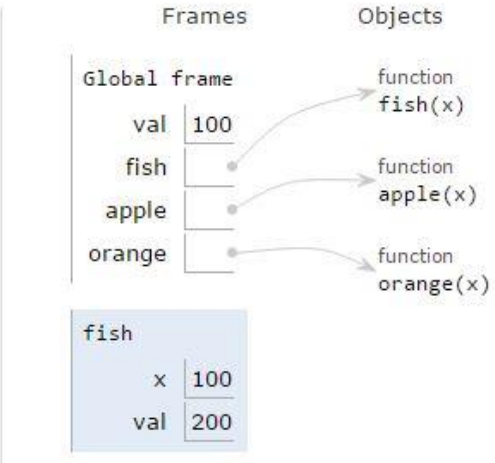
c) **How many different stack frames** (environment frames) are active when the call stack is DEEPEST/LARGEST? (Hint: The global frame counts as one frame.)

MY ANSWER: **3 stack frames**

```
val = 100
def fish(x):
    val = apple(x)
    return val + orange(val)

def apple(x):
    return x * 2

def orange(x):
    return x - 3

print fish(val)
```

ANSWER: **Note:** `orange(val)` **has been called and has returned. Its return value is about to be added to** `val` **and returned by** `fish`**. The deepest the stack will be is when** `fish` **is in the middle of calling** `apple` **or** `orange`**, resulting in 3 stack frames total.**





11) [16 pts] Write a function `max_in_col(pixel_grid)` that, given a list of lists of integers (a `pixel_grid` as in HW3), creates a list of integers that includes the maximum value found in each column of `pixel_grid`. You can assume that `pixel_grid` will always contain at least one row and one column and that the values in `pixel_grid` will be between 0 and 255 and that each row will contain the same number of columns.

Here are a few examples. After the following code is executed:

```
col_list = max_in_col(pixel_grid)
```

If `pixel_grid` contains:	`col_list` will be:

`[ [ 4,  2,  3],`
`[16,  5,  0],`
`[ 3, 200,  6],`
`[ 0,  10, 12] ]`
	`[ 16, 200, 12 ]`

`[ [ 4 ],`
`[ 16 ],`
`[ 3 ],`
`[ 0 ] ]`
	`[ 16 ]`

`[ [ 4,  2,  3] ]`	`[ 4,  2,  3 ]`
`[ [6] ]`	`[ 6 ]`

a) Write TWO assert statements that can be used to test if your function is correct.

Many possible correct answers, here are two examples:

```
assert max_in_col([[6]]) == [6]
assert max_in_col([[1, 2], [122, 5], [0, 255]]) == [122, 255]
```

b) **Fill in your code on the next page →**

```
def max_in_col(pixel_grid):
    """
    Given a list of lists of integers in the range 0 to 255,
    return a list containing the maximum integer in each column
    of the grid.
    You may assume that pixel_grid will be a list containing at
    least one list, containing at least one value.
    """
    # Your code starts here
    result = []
    # For each column, find its max, append to result
    for col in range(len(pixel_grid[0])):
        max_val = 0
        # Find max value over all rows, in this column
        for row in range(len(pixel_grid)):
            if pixel_grid[row][col] > max_val:
                max_val = pixel_grid[row][col]
        result.append(max_val)
    return result
```

OR

```
    result = []
    # Copy first row as current "max" col values
    for col in range(len(pixel_grid[0])):
        result.append(pixel_grid[0][col])

    for row in range(1, len(pixel_grid)):
        for col in range(len(pixel_grid[0])):
            if result[col] < pixel_grid[row][col]:
                result[col] = pixel_grid[row][col]
    return result
```

12) [14 pts] You have a list of dictionaries with the following general structure:

```
presidents = [

{'name':'George Washington', 'vp':'John Adams'},
{'name':'John Adams', 'vp':'Thomas Jefferson'},
{'name':'Zachary Taylor', 'vp':'Millard Fillmore'},
{'name':'Dwight D. Eisenhower', 'vp':'Richard Nixon'},
{'name':'Richard Nixon', 'vp':'Spiro Agnew'},
{'name':'Richard Nixon', 'vp':'Gerald Ford'},

....]
```

For example, George Washington was a president who had John Adams serve as his Vice President ("vp"). Note that if a president served for more than one term or had multiple vice presidents, there could be multiple dictionaries listed for that president.

For this question you need to complete **<u>three</u>** functions.

a) `pres_set(presidents)` will return the set of all presidents. E.g. if `presidents` consists of <u>only</u> the dictionaries listed above:

```
pres_set(presidents)
```

returns the set {`'George Washington', 'John Adams', 'Zachary Taylor', 'Dwight D. Eisenhower', 'Richard Nixon'`}

b) `vp_set(presidents)` will return the set of all vice presidents. E.g. if `presidents` consists of <u>only</u> the dictionaries listed above:

```
vp_set(presidents)
```

returns the set {`'John Adams', 'Thomas Jefferson', 'Millard Fillmore', 'Richard Nixon', 'Spiro Agnew', 'Gerald Ford'`}

c) `promoted_vp_set(presidents)` will return the set of all vice presidents that also served as president. E.g. if `presidents` consists of <u>only</u> the dictionaries listed above:

```
promoted_vp_set(presidents)
```

returns the set {`'John Adams', 'Richard Nixon'`}

Fill in your code on the next page.

```
def pres_set(presidents):
    """ Return a set of all Presidents."""
    # Your code starts here
    prezs = set()
    # Build a set of all presidents
    for pres_rec in presidents:
        prezs.add(pres_rec['name'])
    return prezs


def vp_set(presidents):
    """ Return a set of all Vice Presidents."""
    # Your code starts here
    vps = set()
    # Build a set of all vps
    for pres_rec in presidents:
        vps.add(pres_rec['vp'])
    return vps


def promoted_vp_set(presidents):
    """Returns a set of Vice Presidents who also served as
    Presidents.
    """
    # Your code starts here
    return vp_set(presidents) & pres_set(presidents)

OR

    vps = vp_set(presidents)
    prezs = pres_set(presidents)
    promoted = set()

    for vp in vps:
       if vp in prezs:
           promoted.add(vp)

    return promoted
```