Name: ____**Sample Solution**_____

Email address (UW NetID): _____

CSE 160 Winter 2017: 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 11 questions and 9 pages (including this one), totaling 80 points. Before you start, please check your copy to make sure it is complete. Turn in all 9 pages of the exam, together, when you are finished. When time has been called you must put down your pencil and stop writing. A syntax sheet will be provided separately. **Points will be deducted from your score if you are writing after time has been called.**

Good Luck! Total: 80 points. Time: 50 minutes.

Problem	Points Possible
1	6
2	4
3	5
4	4
5	12
6	4
7	4
8	5
9	12
10	12
11	12
Total	80

1) [6 pts] For each of the if statements below, write the output when x = 10, x = 35, and x = 70 in the table below:

```
a)
if x > 20:
    print "line 1"
elif x > 50:
    print "line 2"
if x < 30:
    print "line 3"

b)
if x < 50:
    print "line 1"
    if x < 50:
        print "line 2"
else:
    print "line 3"
```

	X = 10	X = 35	X = 70
Code a)	**line 3**	**line 1**	**line 1**
Code b)	**line 1** **line 2**	**line 1** **line 2**	**line 3**

2) [4 pts] Write the output of the code below in the box here:

```
sum = 0
for x in range(2, 6):
    for y in range(x, 0, -1):
        sum = sum + 1
print 'sum:', sum
```

MY ANSWER:

sum: 14

3) [5 pts]Given the following variables that have been defined. What is the **result** and **type** of the following expressions (if it is an Error indicate that, and leave result and type blank):

```
x = 5
y = 11
z = 2.5
foo = True
bar = "false"
```

	Result	Type	Error? (yes/no)
`foo and x < y`	**True**	**bool**	**no**
`x / y`	**0**	**int**	**no**
`y >= 0`	**True**	**bool**	**no**
`z / x`	**0.5**	**float**	**no**
`bar[0]`	**"f"**	**string**	**no**

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

```
from operator import itemgetter

data = [ ("Fred", 3, 5), ("Zeke", 5, 3), ("Sam", 5, 6),
         ("Mary", 3, 5), ("Ann", 7, 8) ]

def some_key(x):
    return len(x[0])

print sorted(data, key=some_key)
print sorted(data, key=itemgetter(2), reverse=True)
```

MY ANSWER:

```
[('Sam', 5, 6), ('Ann', 7, 8), ('Fred', 3, 5), ('Zeke', 5, 3), ('Mary', 3, 5)]

[('Ann', 7, 8), ('Sam', 5, 6), ('Fred', 3, 5), ('Mary', 3, 5), ('Zeke', 5, 3)]
```

5) [12 pts] For each of the following statements, show what is printed. Put ----- if nothing is printed.

```
def foo(a):
    if (len(a) % 2 == 0):
        return a
    else:
        return a + "foo"

def bar():
    print "bar"

def cat(c):
    bar()
    c = c + "cat"
    return c
```

a) `print bar()`

bar
None

b) `print cat("red")`

bar
redcat

c) `print foo("green")`

greenfoo

d) `print foo(cat("blue"))`

bar
bluecatfoo

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

```
my_dict = {"pie":[1, 2], "cake":[3], "candy":[7, 6], "salt":[4]}
```

a) `my_dict["cake"]`

[3]

b) `my_dict[3]`

KeyError: 3

c) `my_dict["candy"][0]`

7

d) `my_dict["pie"].append(9)`

None

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

```
odds = {1, 3, 5, 7, 9}
threes = {3, 6, 9}
```

a) `print odds & threes`

set([9, 3])

b) `print odds | threes`

set([1, 3, 5, 6, 7, 9])

c) `print odds - threes`

set([1, 5, 7])

d) `print threes.remove(3)`

None

Note: This notation was also fine: { 3, 9 } and since sets are not ordered, any ordering of elements within the set was fine.

8) [5 pts] What output is produced after running the following piece of code?

```
A = [1, 2]
B = list(A)
C = A
D = A[:]

E = A.append(3)
B.append(A)
D.append(E)
B[1] = 7

print A
print B
print C
print D
print E
```

MY ANSWER:

[1, 2, 3]

[1, 7, [1, 2, 3]]

[1, 2, 3]

[1, 2, None]

None

9) [12 pts] Write a function called `find_value` that takes a **list of lists** and a **value** as arguments and returns a **list** giving the indices of the (sub)list and position in that list where the first instance of the value was found. If the value does not exist in any of the (sub)lists, the function returns [-1, -1]. For example, given:

```
my_list = [ [2, 5], [1, 3, 4], [4], [7, 3] ]
```

the call: `find_value(my_list, 4)` would return: `[1, 2]` and
the call: `find_value(my_list, 7)` would return: `[3, 0]` and
the call: `find_value(my_list, 8)` would return: `[-1, -1]`

```
def find_value(lst, val):

  # Write your code here
  for sub_num in range(len(lst)):
      for num in range(len(lst[sub_num])):
          if lst[sub_num][num] == val:
              return [sub_num, num]
  return [-1, -1]
```

Another solution:

```
   for sub_num in range(len(lst)):
      if val in lst[sub_num]:
          return [sub_num, lst[sub_num].index(val)]
  return [-1, -1]
```

10) [12 points] Write a function called `num_to_names_map` that takes a dictionary `dict` as an argument. `dict` maps a name to a set of that person's favorite numbers. For example:

```
fav_nums = {"fred":{1, 2}, "mary":{2, 3}, "jim":{7, 1, 3}}
```

The function should return a dictionary mapping numbers to a set of people who have that number as one of their favorites. For example:

```
print num_to_names_map(fav_nums)
```

Would print something like this:

```
{1: set(['jim', 'fred']), 2: set(['mary', 'fred']), 3: set(['jim', 'mary']),
 7: set(['jim'])}
```

It may be useful to remember that you can use a for loop to iterate through sets.

```
def num_to_names_map(dict):

    # Write your code here
    new_dict = {}
    for name in dict.keys():
        for num in dict[name]:
            if num in new_dict.keys():
                new_dict[num].add(name)
                #OR new_dict[num] = new_dict[num] | set([name])
            else:
                new_dict[num] = {name}
    return new_dict
```

11) [12 pts] a) **Draw** the entire environment, including all active environment frames and all user-defined values, **at the moment that the MINUS OPERATION IS performed**. Feel free to draw out the entire environment, but be sure to CLEARLY indicate what will exist at the moment the **MINUS** 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:	25

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:	4

```
x = 10
y = 200

def pig(x):
    return x + 5

def chicken(x):
    temp = pig(x)
    return koala(temp) - temp

def koala(y):
    return pig(y) + y

print chicken(pig(x))
```

MY ANSWER: