Name: _____________________________________

Email address: _____________________________________

Quiz Section: __________

CSE 140 Winter 2014: Quiz

(closed book, closed notes, no calculators)

Instructions: This exam is closed book, closed notes. It contains 3 questions and 4 pages (including this one), totaling 10 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: 10 points.

<table>
<thead>
<tr>
<th>Page</th>
<th>Max Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>
1) [3 pts] You are given a dictionary of dictionaries (as shown below), that maps pollsters to stateEdges (remember that a stateEdge is a dictionary that maps states to edges). Write a function `wa_edges` that returns a list of tuples where each tuple holds the name of the pollster as the first element and the edge corresponding to WA (Washington) as the second element. If that pollster does not have an edge for WA, store its value as None.

```python
def wa_edges(data):
    ''' Given a dictionary that maps pollsters to stateEdges, return a list of tuples containing the pollster's name and its corresponding edge for 'WA'. If there is no edge specified for WA, use None as the edge value.'''
    # Your code starts here
```

```python
input_data = { "Gallup": { "WA": 7, "CA": 15, "UT": -30 },
              "SurveyUSA": { "CA": 14, "CO": 2, "CT": 13, "FL": 0 },
              "Omniscient": { "AK": -14.0, "WA": -2.3, "CA": 20.9 } }
```

For example, calling `wa_edges(input_data)` returns a list containing these tuples (the order of the tuples may differ):

```python
[ ("Gallup", 7), ("SurveyUSA", None), ("Omniscient", -2.3) ]
```
2) [3 pts]: Given a dictionary of dictionaries (as used in the previous problem), write a function `pollster_states` that returns a dictionary that maps pollsters to a list of the states shown in their associated stateEdge.

For example, calling `pollster_states(input_data)` returns a new dictionary containing these values (the order of the values may differ):

```python
{ "Gallup": [ "WA", "CA", "UT" ],
  "SurveyUSA": [ "CA", "CO", "CT", "FL" ],
  "Omniscient": [ "AK", "WA", "CA" ]
}
```

def pollster_states(data):
    ''' Given a dictionary that maps pollsters to stateEdges, return a dictionary that maps each pollster to a list of the states shown in that Pollster's corresponding stateEdge.'''
    # Your code starts here
3) [4 pts] a) **Draw** the entire environment, including all active environment frames and all user-defined variables, **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. by crossing out things that will not exist).

b) 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:**

c) What is printed?

**MY ANSWER:**

def bar(x):
    return x + 5

def foo(x):
    result = bar(x)
    result = bar(result)
    return result

def zippy(y):
    return bar(foo(y) - 3)

print zippy(2)