CSE 160 Section 6 Problems

Use the following function and list to help answer problems 1 and 2:

```
def row_to_edge(row):
    """
    Returns the difference between the "Dem" and "Rep"
     values of row.
    """
    return float(row["Dem"]) - float(row["Rep"])

rows = [ { "State":"AK", "Dem":"41.3", "Rep":"55.3" },
    { "State":"AL", "Dem":"38.4", "Rep":"60.7" },
    { "State":"AR", "Dem":"36.9", "Rep":"60.5" },
    ... ]
```

1. Given the function `row_to_edge` and a list of rows, print the number of Democratic states, Republican states, and neutral states in the list. If `row_to_edge` returns a positive `float`, then that state is considered to be a Democratic state, if it returns a negative `float` then the state is considered to be a Republican state, and if it returns `0` the state is considered to be a neutral state.

2. Print the “most Democratic” state and the “most Republican” state. Most Democratic is defined as the state with the highest value returned by the row_to_edge function, and most Republican is defined as the state with the lowest value returned by the row_to_edge function.

Use the following dictionary of dictionaries to help answer problems 3, 4, and 5:

```
data = { "Gallup": { "WA": 7, "CA": 15, "UT": -30 },
    "SurveyUSA": { "CA": 14, "CO": 2, "CT": 13, "FL": 0, "KY":
-14 },
    "RAND": { "NY": 11.2, "AZ": -9.8, "AR": -18.9 },
    ... }
```

3. Write one line of code that will execute the following commands:

Print a list of the keys in the dictionary data.

Print all of the key-value pairs in the dictionary data as a list of tuples.

Print all of the keys in the dictionary associated with the pollster `"Gallup"`.

Print the value for `"CA"` in `"RAND"` or `None` if it does not exist.

4. Write a function `get_results_for(data, state)` that returns a list of tuples (pollster, edge). Edge is defined as the difference between the "Dem" and "Rep" values. The first element is the name of the pollster and the second element the edge corresponding to the given state. If the pollster and state do not have an edge, store its value as `None`.

5. (Extra)Write a function that returns the list of tuples (pollster, edge) for California `"CA"`. You should use your function from problem 4.

6. (Extra)Write a function that takes a list as a parameter, and returns a set containing the elements that appear more than once in the list.
For example: `duplicates([1, 3, 2, 4, 3, 1, 1])` returns `set([1, 3])`.

7. What's the output for the following code?

```
from operator import itemgetter
fruits = ["watermelons", "oranges", "kiwis", "grapes"]
print (sorted(fruits, key = len))
print (sorted(fruits))
prices = [("watermelons", 23), ("oranges", 12), ("kiwis",
30), ("grapes", 25)]
sorted_by_name = sorted(prices, key=itemgetter(0))
print(sorted_by_name)
sorted_by_price = sorted(sorted_by_name, key=itemgetter(1),
reverse=True)
print(sorted_by_price)
```

CSE 160 Section 6 Solutions

1. Print the number of Democratic states, Republican states, and neutral states in the list:

```
num_dem = 0
num_rep = 0
num_neutral = 0
for row in rows:
    edge = row_to_edge(row)
    if edge > 0:
        num_dem += 1
    elif edge < 0:
        num_rep += 1
    Else:
        num_neutral += 1
print("Democratic States:", num_dem)
print("Republican States:", num_rep)
print("Neutral States:", num_neutral)
```

2. Print the “most Democratic” state and the “most Republican” state:

```
most_dem_state = None
most_rep_state = None
min_edge = math.inf
max_edge = -math.inf
for row in rows:
    edge = row_to_edge(row)
    if edge > max_edge:
        max_edge = edge
        most_dem_state = row["State"]
    if edge < min_edge:
        min_edge = edge
        most_rep state = row[“State"]
print("Most Democratic state: " + most_dem_state)
print("Most Republican state: " + most_rep_state)
```

3.

(a) print(data.keys())
(b) print(data.items())
(c) print(data["Gallup"].keys())
(d) print(data["RAND"].get("CA", None))

4. Write a function get_results_for(data, state) that returns a list of tuples (pollster, edge):

```
def get_results_for(data, state):
    """
    Given a dictionary of pollsters mapped to rows, returns a list
    of tuples containing the pollster’s name and it’s corresponding
    edge for state. If there is no edge specified for state, stores None.
    """
    results = []
```

```
        for pollster in data:
            if state in data[pollster].keys():
                pollster_edge = (pollster, data[pollster][state])
            else:
                pollster_edge = (pollster, None)
            results.append(pollster_edge)
        return results
```

5. Write a function that returns the list of tuples for California "CA":

```
    def california_results(data):
        return get_results_for(data, "CA")
```

6.

```
def duplicates(input_list):
    seen = set()
    result = set()
    for element in input_list:
        if element in seen:
            result.add(element)
        seen.add(element)
    return result
```

7.

```
['kiwis', 'grapes', 'oranges', 'watermelons']
['grapes', 'kiwis', 'oranges', 'watermelons']
[('grapes', 25), ('kiwis', 30), ('oranges', 12), ('watermelons', 23)]
[('kiwis', 30), ('grapes', 25), ('watermelons', 23), ('oranges', 12)]
```

Name (first and last):

Email:

---

a) What's the thing you find most confusing about itemgetter?

b) Any questions or general feedback for your section TA?

Name (first and last):

Email:

---

a) What's the thing you find most confusing about itemgetter?

b) Any questions or general feedback for your section TA?