COMP 53: Containers Lab, part 3

Instructions: In this lab, we are going to review pairs and maps.

- Get into groups of **at most two people** to accomplish this lab.
- At the top of your source code files list the group members as a comment.
- Each member of the group must individually submit the lab in Canvas.
- This lab includes **35 points** in aggregate. The details are given in the following.

1 `city.h`

Modify `city.h` from the previous lab according to the following:

1. Include `utility` library.
2. Rather than having two protected data components, a `City` object may have a single data component `namePopulationPair` which is a pair of city name and its population ***(2 points)***.
3. The default constructor receives the two inputs as name and population (similar to previous case), and initializes the `namePopulationPair` ***(2 points)***.
4. `setName(...)` sets the name part of `namePopulationPair` ***(2 points)***.
5. `setPopulation(...)` sets the population part of `namePopulationPair` ***(2 points)***.
6. `getName()` returns the name part of `namePopulationPair` ***(2 points)***.
7. `getPopulation()` returns the population part of `namePopulationPair` ***(2 points)***.
8. `printInfo()` is the same as before.

2 `main.cpp`

In `main.cpp` do the following step by step:

1. Include `map` as well as `city.h`.
2. Globally define array `cityArray[]` consisting of cities with the following details:
   (a) Los Angeles with population of 4 million
   (b) San Diego with population of 1.5 million
   (c) San Francisco with population of 900 thousand
   (d) Sacramento with population of 500 thousand
   (e) Stockton with the population of 300 thousand
3. Globally define a map from strings to unsigned integers, without initial values. Call it `cityMap` ***(1 points)***.
4. Define the following functions on maps. Pass all maps to these functions as *reference*. Moreover, annotate the input set to be *constant*, if the function is not supposed to modify that input map. In addition, use *range-based `for` loops* to traverse the maps.

   Note: When doing range-based `for` loop, each entry of map is essentially a *pair*.

(a) Define function `void initMap(...)` that receives a map from strings to unsigned integers, an array of elements of type `City` as a second input, and an integer as its third input. The third input represents the number of elements in the input array. Initialize the input map with the elements existing in the input array, by setting the key/value components of each map entry ***(3 points)***.

(b) Define function `void printCityMap(...)` that receives a map from strings to unsigned integers as input and prints the key/value entries within that map in the standard output ***(3 points)***.

(c) Define function `unsigned int findPopulation(...)` that receives a map from strings to unsigned integers, along with a name of a city (a string). It returns the population of the city if there is a match. Otherwise, it returns 0 ***(3 points)***.

(d) Define function `void updatePopulation(...)` that receives a map from strings to unsigned integers along with a name of city (a string) and the population of that city (an unsigned integer). If there is already an entry for the input city name, the population is updated within the map according to the input population. Otherwise, a new entry is created for the input city name and population within the input map ***(3 points)***.

(e) Define function `void removeFromCityMap(...)` that receives a map from strings to unsigned integers as input, along with a name of a city (a string). It removes the entry associated with that city name. Avoid iterating on map entries ***(3 points)***.

In `main()` function do the following step by step, using the functions defined above:

(i) Initialize `cityMap` according to array `cityArray[]` using the function defined above ***(1 points)***.

(ii) Print out the entries of `cityMap`, using the appropriate function defined above ***(1 points)***.

(iii) Find population of Sacramento within `cityMap` using the function defined above, and print out the result ***(1 points)***.

(iv) Find population of Chicago within `cityMap` using the function defined above, and print out the result ***(1 points)***.

(v) Update the population of Stockton to 350000 within `cityMap` and then find its population within that map. Both functions are defined above ***(1 points)***

(vi) Update the population of Portland to 700000 within `cityMap` and then find its population within that map. Both functions are defined above ***(1 points)***

(vii) Remove San Francisco's entry from `cityMap`, and then print out the entries of this map ***(1 points)***.

The output of the program may look like the following:

```
Initializing cityMap with cityArray[]:
Los Angeles: 4000000
Sacramento: 500000
San Diego: 1500000
San Francisco: 900000
Stockton: 300000

Find population of Sacramento: 500000
Find population of Chicago: 0

Find population of Stockton after update: 350000
Find population of Portland after update: 700000

Removing San Francisco from cityMap:
```

```
Los Angeles: 4000000
Portland: 700000
Sacramento: 500000
San Diego: 1500000
Stockton: 350000
```