COMP 53: Objects and Classes Lab, part 2

Instructions: In this lab, we are going to review objects and classes.

- Get into groups of **at most two people** to accomplish this lab.
- At the top of your source code list the group members as a comment.
- Each member of the group must individually submit the lab in Canvas.
- This lab includes **34 points** in aggregate. The details are given in the following.

1 Class `City`

First, let's define class `City` in header file `city.h`, with the following details.

1. Each `City` has the following **data components**:
   - The name of the city: `name` as a string, and
   - The population of the city: `population` as an unsigned integer.

   Each of the aforementioned data components must be **hidden** from the class user ***(2 points)***.
2. Define setter function `void setName(string name)` in inlined form. Since the function parameter has the same name as the data component `name`, you need to use `this` pointer to refer to the data component `name` within the body of the function ***(2 points)***.
3. Define setter function `void setPopulation(unsigned int population)` in inlined form. Since the function parameter has the same name as the data component `population`, you need to use `this` pointer to refer to the data component `population` within the body of the function ***(2 points)***.
4. Define getter functions `getName` and `getPopulation` accordingly in inlined form ***(4 points)***.
5. Define default constructor `City()` that sets `name` to `N/A`, and `population` to `0`, in inlined form ***(2 points)***.
6. Define a second constructor `City(string name, int population)` that sets data components `name` and `population` using the input arguments. Again, since the parameters and data components have the same name, you need to use `this`. Define this constructor in inlined form, as well ***(2 points)***.

2 Class `Cities`

Second, let's define class `Cities` in header file `cities.h` and source file `cities.cpp`. The header file includes the definition of the class, and the source file includes the definition of class functions. The details are as follows:

1. Each `Cities` object includes a vector of `City` objects. Name this vector `cityList`, and make it hidden to class users. Note that you need to include `city.h` header file in `cities.h`, in order to use class `City` ***(2 points)***.
2. List the two public functions for this class `void readCities()` and `void printCityList()` in the definition of class `Cities` ***(2 points)***.

   In `cities.cpp` file, define the functions listed in the previous step. In this file, you must include file `cities.h`. Details are as follows:

3. Function `readCities()` iteratively reads city name and population from standard input and puts them in the `cityList`. Continue reading city information until user enters `X` ***(3 points)***.

4. Function `printCityList()` iterates through the vector `cityList` and prints the name and population of each city, separated by `':'` ***(3 points)***.

3 Class `State`

Consider the following definition of class `State` as the starting point.

```
class State {
        public:
                State() {name = "N/A";}
                void setName(string stateName) {name =  stateName;}
                string getName() const {return name;}

        private:
                string name;

};
```

Put this definition in source file `state.cpp` and extend it as follows:

1. Add a private `Cities` data component named `stateCities` to this class. For it to work, you must include file `cities.h` in `state.cpp` ***(2 points)***.

2. Add `void readStateCities()` as a public function of this class. This function invokes `stateCities`'s `readCities()` function to fill up the list ***(2 points)***.

3. Add `void printStateCities()` as a public function of this class. This function invokes `stateCities`'s `printCityList()` function to print the list ***(2 points)***.

4 `Main` function

Define `main` function in `state.cpp` that does the following step by step.

1. Reads the state name from standard input ***(1 points)***.

2. Creates a state and sets its name to the entered name ***(1 points)***.

3. Reads the list of city names and populations by calling `readStateCities()` ***(1 points)***.

4. Prints the list of city names and populations by calling `printStateCities()` ***(1 points)***.

The output of the program may look like the following:

```
State: California
Enter city name and population. To end, enter X.
City: LosAngeles
Population: 4000000
City: SanFrancisco
Population: 900000
City: Sacramento
Population: 500000
City: Stockton
Population: 300000
City: X
```

```
The state California has the following cities/populations:
LosAngeles: 4000000
SanFrancisco: 900000
Sacramento: 500000
Stockton: 300000
```