CS 172 Exam 2 (Spring 2017) - SimCity

You are allowed to look at any of your previous programming solutions, slides, notes, or book for the exam. You may also look at http://www.cplusplus.com/reference/string/string/. You will have until midnight to turn in this exam. I am available on email to answer any clarification questions until 2pm today. If you have any questions about what to create for this exam please ask me during this time.

We need to work on the honor system. You must put the following statement at the top of any source files you write:

I affirm that all code given below was written solely by me, <give your name>, and that any help I received adhered to the rules stated for this exam.

Please save your solution to GitHub repo named CS172-Exam2, and submit the GitHub URL to me via email. Please note, that submission to GitHub is 5% of your score. So you will lose 5% if you do not submit via GitHub.

You are going to write SimCity. Well actually it is something much smaller. What matters in this SimCity is the favorite colors for the citizens of the city. You will be creating a City class that has a city name and population. You will also be creating a Citizen class that will have an id, first name, last name, and favorite color.

Define a class called **Citizen** and put this in a class definition file called **Citizen.h**.

- The class has the following **public methods** (behaviors) implemented in a file called **Citizen.cpp**:

```
class Citizen
{
public:

    //Creates a new Citizen object with the passed in
    //id, name, and favorite color
    //Once a citizen is created, you can't change their id or name
    //but you can change their favorite color
    Citizen(int id, string firstName, string lastName, string color);

    //Creates a new Citizen object by copying data from the
    //passed in citizen
    Citizen(Citizen* citizen);

    //Returns the first name of this citizen
    string getFirstName();

    //Returns the last name of this citizen
    string getLastName();

    //Returns the id for this citizen
    int getId();

    //Returns the favorite color for this citizen
    string getFavoriteColor();

    //Sets the favorite color for this citizen
    void setFavoriteColor(string color);
};
```

- Think about what additional **private properties** the **Citizen** class needs to have in order to correctly implement the above behaviors.

Also define a class called **City**, and put this class definition in **City.h**.

- The class has the following **public methods** (behaviors) implemented in a file called **City.cpp**:

```
class City
{

public:

    //Creates a new city with the given name
    //When the city is created you need to restore
    //it's population from a file.
    //Hint: You will want to make the file name
    //be based on the name of the city.
    City(string cityName);

    //This is the destructor for the city.  When
    //this city is destroyed, save the population of
    //the city to a file so that you can restore
    //it in the constructor the next time that
    //a city with this name is created.
    ~City();

    //Returns the city name
    string getCityName();

    //Returns the number of citizens in this city
    int populationSize();

    //Returns the citizen at the given index.
    Citizen* getCitizenAtIndex(int index);

    //Adds a citizen to this city. You will need to
    //make a copy of this citizen so that you make
    //sure to keep it around for the life of the city.
    void addCitizen(Citizen* citizen);

    //Returns the citizen with the given id.
    Citizen* getCitizenWithId(int id);

    //Returns a vector of citizens that all have
    //the given color as their favorite color.
    //For example, if color is “Blue” this will return all citizens
    //for this city who’s favorite color is Blue.
    vector<Citizen*> getCitizensForFavoriteColor(string color);
};
```

- Think about what additional **private properties** the **City** class needs to have in order to correctly implement the above behaviors.

More Information / Hints

- You can assume that the first name for all citizens is a single word.
- You can assume that the last name for all citizens is a single word.
- You can assume that favorite colors for all citizens is a single word.
- In your constructor for the City you need to open a file for that city and restore the citizens for the city. I would recommend that you use the name of the city in the name of your file (ie spokane.txt). You would use something like `string fileName = cityName + ".txt";` to do this.
- In the destructor for City you need to save the population for that city to the file for that city.
- When you write the citizen data to the file you can do something like one citizen per line and then on each line output the citizen information with a space between each piece of data. For example

```
1 Stephen Johnson Green
2 Joe Miller Blue
```

- The test code for this SimCity uses new and delete to allocate Citizen object. Once the code below adds a citizen to the city it deletes the citizen because it no longer needs that object. This means you will need to make a copy of the citizen in your addCitizen method.
- Be sure that you have a delete for every new in your own code.

Testing your Solution

Use test code below to test your solution. You should not change this test code, just use it. You will need to run this code three times in a row to do a full test to make sure you are correctly saving and restoring the population for your city. Each time you run the code you should see that that error count is 0. The 2nd and 3rd time that you run the code you should see it output SUCCESS.

The first run should have this
ERROR COUNT: 0.
Be sure to run this 3 times and be sure that on the 2nd and 3rd run you get 1 SUCCESS.

The second run should have this
SUCCESS: Found Richard
ERROR COUNT: 0.
Be sure to run this 3 times and be sure that on the 2nd and 3rd run you get 1 SUCCESS.

The third run should have this
SUCCESS: Found Richard
SUCCESS: Richard's color successfully changed!
ERROR COUNT: 0.
Be sure to run this 3 times and be sure that on the 2nd and 3rd run you get 1 SUCCESS.

Once your classes are designed and implemented, test it with the following program:

```
//
//  main.cpp
//  CS172-Exam2
//
//

#include <iostream>
#include "Citizen.h"
#include "City.h"

using namespace std;

int main() {

    /*
     * DO NOT CHANGE THIS CODE FOR YOUR EXAM!
     */

    int errors = 0;

    //Create the cities
    City* katchem = new City("Katchem");
    City* spokane = new City("Spokane");
    City* seattle = new City("Seattle");

    //Check if we need to add citizens to Spokane
    //Bonus point if you can tell me why I picked these names and Ids ;)
    if (spokane->populationSize() == 0)
    {
        Citizen* brandon = new Citizen(1, "Brandon", "Semenuk", "Green");
        spokane->addCitizen(brandon);
        delete brandon;

        Citizen* antoine = new Citizen(2, "Antoine", "Bizet", "Blue");
        spokane->addCitizen(antoine);
        delete antoine;

        Citizen* carson = new Citizen(3, "Carson", "Storch", "Pink");
        spokane->addCitizen(carson);
        delete carson;
```

```
    Citizen* kurt = new Citizen(4, "Kurt", "Sorge", "Blue");
    spokane->addCitizen(kurt);
    delete kurt;

    Citizen* kyle = new Citizen(5, "Kyle", "Strait", "Pink");
    spokane->addCitizen(kyle);
    delete kyle;
}

//Check if we need to add citizens to Katchem
//Bonus point if you can tell me why I picked these names ;)
if (katchem->populationSize() == 0)
{
    Citizen* Bartley = new Citizen(6, "Bartley", "Andre", "Green");
    katchem->addCitizen(Bartley);
    delete Bartley;

    Citizen* Daniel = new Citizen(7, "Daniel", "Coster", "Blue");
    katchem->addCitizen(Daniel);
    delete Daniel;

    Citizen* Daniele = new Citizen(8, "Daniele", "DeLuliis", "Blue");
    katchem->addCitizen(Daniele);
    delete Daniele;

    Citizen* Richard = new Citizen(9, "Richard", "Howarth", "Pink");
    katchem->addCitizen(Richard);
    delete Richard;

}
else
{
    Citizen* Richard = katchem->getCitizenWithId(9);
    if (Richard == NULL)
    {
        cout << "ERROR: Could not find Richard" << endl;
    }
    else
    {
        cout << "SUCCESS: Found Richard" << endl;
    }

    if (Richard->getFavoriteColor() == "Pink")
    {
        Richard->setFavoriteColor("Purple");
    }
    else if (Richard->getFavoriteColor() == "Purple")
    {
        cout << "SUCCESS: Richard's color successfully changed!" << endl;
    }
    else
    {
        errors++;
        cout << "ERROR: Richard's color is wrong. " << endl;
    }
}

if (seattle->populationSize() == 0)
{
    //We are going to add 1000's of citizens to seattle
    for (int i = 0; i < 10000; i++)
    {
        string firstname = "first" + to_string(i);
        string lastname = "last" + to_string(i);
        Citizen* c = new Citizen(1000+i, firstname, lastname, "Green");
        seattle->addCitizen(c);
        delete c;
    }
}

if (katchem->populationSize() != 4)
```

```
    {
        errors++;
        cout << "ERROR: Katchem has the wrong population size" << endl;;
    }

    if (spokane->populationSize() != 5)
    {
        errors++;
        cout << "ERROR: Spokane has the wrong population size" << endl;;
    }

    if (seattle->populationSize() != 10000)
    {
        errors++;
        cout << "ERROR: Seattle has the wrong population size" << endl;;
    }



    vector<Citizen*> blueLovers = spokane->getCitizensForFavoriteColor("Blue");
    if (blueLovers.size() != 2)
    {
        errors++;
        cout << "ERROR: wrong number of blue lovers in Spokane " << endl;
    }

    vector<Citizen*> greenLovers = katchem->getCitizensForFavoriteColor("Green");
    if (greenLovers.size() != 1)
    {
        errors++;
        cout << "ERROR: wrong number of green lovers in Katchem " << endl;
    }
    else
    {
        Citizen* greenlover = greenLovers.at(0);
        if (greenlover->getLastName() != "Andre" && greenlover->getLastName() != "Bartley")
        {
            errors++;
            cout << "ERROR: The green lover was wrong" << endl;
        }
    }

    cout << "ERROR COUNT: " << errors << ".\n";
    cout << "Be sure to run this 3 times and be sure that on the 2nd and 3rd run you get 1
SUCCESS." << endl;


    //Delete our cities when we are done with them
    delete katchem;
    delete spokane;
    delete seattle;

    return 0;
}
```

Grade Rubric

Category	Percent
Correct use of member variables and methods.	10%
Correctly read and write city data to file.	15%
Correct Implementation for Citizen.	20%
Correct Implementation for City.	20%
Comments	20%
Successful GitHub submission	5%
Correct use of new and delete	10%