COMP 53: Containers Lab, part 2

Instructions: In this lab, we are going to review lists.

- 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 **36 points** in aggregate. The details are given in the following.

1 city.h

Create `city.h` with the following content.

```
#ifndef CITY_H
#define CITY_H

#include<string>

class City {
        public:
                City(string nm, unsigned int pop) {
                        name = nm;
                        population = pop;
                }
                void setName(string name) {this -> name = name;}
                void setPopulation(unsigned int population)
                        {this -> population = population;}
                string getName() const {return this-> name;}
                unsigned int getPopulation() const {return this -> population;}
                virtual void printInfo() const {
                        cout<<getName()<<": "<<getPopulation()<<endl;
                }
        protected:
                string name;
                unsigned int population;
};

#endif
```

2 main.cpp

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

1. Include `list` as well as `city.h`.
2. Globally define array `a[]` of integers consisting of values: 2,8,1,7, and 3 in order.
3. Globally define array `cityArray[]` of cities 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

4. Globally define a list of integers, without initial values. Call it `intList` ***(1 points)***.

5. Globally define a list of cities, without initial values. Call it `cityList` ***(1 points)***.

6. Define the following functions on lists.

   (a) Define function `void initList(...)` that receives a list of elements of *any arbitrary type* T, an array of elements of type T 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 list with the elements existing in the input array ***(3 points)***. Note that since the list is being modified, you must pass it as reference.

   (b) Define function `void printList(...)` that receives a list of elements of *any arbitrary type* T as input and prints the elements residing within that list in the standard output. Since the input vector is not being modified by this function, define it to be constant. In addition, use range-based `for` loops to traverse the list ***(3 points)***.

   (c) Define a specific printing function for list of cities. Specifically, define function `void printCityList(...)` that receives a list of cities as input and prints the elements residing within that list in the standard output (using `printInfo()` function). Since the input list is not being modified by this function, define it to be constant. In addition, use range-based `for` loops to traverse the list ***(3 points)***.

   (d) Define function `void rotateListRight(...)` that receives a list of elements of *any arbitrary type* T as input along with an integer n. It rotates the input list to right n number of times ***(3 points)***. As an example of how it works check the output of the program below.

   (e) Define function `void rotateListLeft(...)` that receives a list of elements of *any arbitrary type* T as input along with an integer n. It rotates the input list to left n number of times ***(3 points)***. As an example of how it works check the output of the program below.

   (f) Define function `void removeFromCityList(...)` that receives a list of cities as input, along with a name of city (a string). It removes the first occurrence of the city matched with the input city name from the input list. Use iterator-based `for` loops to traverse the list ***(4 points)***. As an example of how it works check the output of the program below.

   (g) Define function `void InsertToCityList(...)` that receives a list of cities as input, along with a `City` object and a name of city (a string). It inserts the input city before the first occurrence of the city whose name matches the city name given as 3rd input. Use iterator-based `for` loops to traverse the list ***(4 points)***. As an example of how it works check the output of the program below.

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

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

(ii) Print out the elements of `intList`, using to the appropriate function defined above ***(1 points)***.

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

(iv) Print out the elements of `cityList`, using to the appropriate function defined above ***(1 points)***.

(v) Rotate `intList` to right for 2 steps, and print out the resulting list ***(1 points)***.

(vi) Rotate `cityList` to right for 1 step, and print out the resulting list ***(1 points)***.

(vii) Rotate `intList` to left for 1 step, and print out the resulting list ***(1 points)***.

(viii) Rotate `cityList` to left for 2 steps, and print out the resulting list ***(1 points)***.

(ix) Remove Sacramento from `cityList` and print out the updated list ***(1 points)***.

(x) Insert the city Redding with population 90 thousand to `cityList` sitting before Stockton. Next, print out the updated list ***(2 points)***.

The output of the program may look like the following:

```
Initializing intList with a[]:
2 8 1 7 3

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

Rotating intList to right twice:
7 3 2 8 1

Rotating cityList to right once:
Stockton: 300000
Los Angeles: 4000000
San Diego: 1500000
San Francisco: 900000
Sacramento: 500000

Rotating intList to left once:
3 2 8 1 7

Rotating cityList to left twice:
San Diego: 1500000
San Francisco: 900000
Sacramento: 500000
Stockton: 300000
Los Angeles: 4000000

Removing Sacramento from cityList:
San Diego: 1500000
San Francisco: 900000
Stockton: 300000
Los Angeles: 4000000

Inserting Redding to cityList before Stockton:
San Diego: 1500000
San Francisco: 900000
Redding: 90000
Stockton: 300000
Los Angeles: 4000000
```