COMP 53: Lists Lab, part 5

Instructions: In this lab, we are going to review doubly-linked lists with dummy nodes.

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

1 `city.h` and `citynode.h`

Consider `city.h` and `citynode.h` (doubly-linked nodes of cities) from the previous lab.

2 `citylist.h`

Consider class `CityList` that implements the doubly-linked list of cities with dummy node. It keeps track of the first and last elements of the list (through `head` and `tail` pointers, respectively).

```
#ifndef CITYLIST_H
#define CITYLIST_H

#include<string>
#include "citynode.h"
class CityList {
        public:
                CityList() { head = tail = new CityNode(City());}
                void append(CityNode *cityNode);
                void prepend(CityNode *cityNode);
                void printCityList();
                CityNode *search(string cityName);
                void insert(CityNode *currNode, CityNode *cityNode);
                void remove(CityNode *currNode);
        private:
                CityNode *head;
                CityNode *tail;
};
#endif
```

Note that the dummy node is created upon constructing the list, and `head` and `tail` point to it.

1. Redefine `void append(...)` function considering the fact that `CityList` starts with a dummy node. This would make its definition simpler ***(3 points)***.
2. Redefine `void prepend(...)` function considering the fact that `CityList` starts with a dummy node. This would make its definition simpler ***(3 points)***.
3. Redefine `CityNode *search(...)` function considering the fact that `CityList` starts with a dummy node. This would make its definition slightly different ***(3 points)***.
4. Redefine `void printCityList()` function considering the fact that `CityList` starts with a dummy node. This would make its definition slightly different ***(3 points)***.
5. Redefine `void insert(...)` function considering the fact that `CityList` starts with a dummy node. This would make its definition simpler ***(3 points)***.
6. Redefine `void remove(...)` function considering the fact that `CityList` starts with a dummy node. This would make its definition simpler ***(3 points)***.

3 main.cpp

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

1. Globally define array `cityArray[]` consisting of cities with the following details:
   (a) Los Angeles with population of 4340174
   (b) San Diego with population of 1591688
   (c) San Francisco with population of 871421
   (d) Sacramento with population of 505628
   (e) Stockton with the population of 323761
   (f) Redding with the population of 90292
   (g) Las Vegas with the population of 711926
   (h) Reno with the population of 289485
   (i) Portland with the population of 730428
   (j) Seattle with the population of 752180
   (k) Eugene with the population of 221452
2. Globally define two `CityLists` named as `cityList1` and `cityList2` ***(1 points)***.
3. Pass `CityLists` to these functions as *reference*.
   (a) Define function `void initCityListByAppend(...)` that receives a `CityList`, 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 `CityList` with the elements existing in the input array, by iteratively invoking `append()` function ***(2 points)***.
   (b) Define function `void initCityListByPrepend(...)` that receives a `CityList`, 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 `CityList` with the elements existing in the input array, by iteratively invoking `prepend()` function ***(2 points)***.

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

(i) Initialize `cityList1` according to array `cityArray[]` by appending, using the function defined above ***(1 points)***.

(ii) Print out the entries of `cityList1`, using the appropriate function defined as part of `CityList` class ***(1 points)***.

(iii) Initialize `cityList2` according to array `cityArray[]` by prepending, using the function defined above ***(1 points)***.

(iv) Print out the entries of `cityList1`, using the appropriate function defined as part of `CityList` class ***(1 points)***.

(v) Add a `CityNode` for city Phoenix with population 1660472 after Stockton's node in `cityList1`. Next, print out the resulting list. *Hint*: You can first search for Stockton, and use the pointer returned by the search function as the current node in insertion function ***(2 points)***.

(vi) Remove the node referring to Reno in `cityList2`. Next, print out the resulting list. *Hint*: You can first search for Reno, and use the pointer returned by the search function as the current node in removal function ***(2 points)***.

The output of the program may look like the following:

```
Initializing cityList1 with cityArray[] using appending:
Los Angeles: 4340174
San Diego: 1591688
San Francisco: 871421
Sacramento: 505628
Stockton: 323761
Redding: 90292
Las Vegas: 711926
Reno: 289485
Portland: 730428
Seattle: 752180
Eugene: 221452

Initializing cityList2 with cityArray[] using prepending:
Eugene: 221452
Seattle: 752180
Portland: 730428
Reno: 289485
Las Vegas: 711926
Redding: 90292
Stockton: 323761
Sacramento: 505628
San Francisco: 871421
San Diego: 1591688
Los Angeles: 4340174

Searching for Stockton in cityList1, and inserting Phoenix after it:
Los Angeles: 4340174
San Diego: 1591688
San Francisco: 871421
Sacramento: 505628
Stockton: 323761
Phoenix: 1660472
Redding: 90292
Las Vegas: 711926
Reno: 289485
Portland: 730428
Seattle: 752180
Eugene: 221452

Searching for Reno in cityList2, and removing that node:
Eugene: 221452
Seattle: 752180
Portland: 730428
Las Vegas: 711926
Redding: 90292
Stockton: 323761
Sacramento: 505628
San Francisco: 871421
San Diego: 1591688
Los Angeles: 4340174
```