

COMP 53: Lists Lab, part 3

Instructions: In this lab, we are going to review doubly-linked 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`

Consider `city.h` from the previous lab.

2 `citynode.h`

Modify `citynode.h` from the previous lab to support two links: a link to the next node, and a link to the previous node. Modify the default constructor accordingly (**3 points**).

Essentially a `CityNode` object is used as an element of the list for cities, which consists of a data component (a city), and a pointer to the next `CityNode`, and a pointer to the previous `CityNode`.

3 `citylist.h`

Modify class `CityList` to implement the doubly-linked list of cities, which keeps track of the first and last elements of the list (through `head` and `tail` pointers, respectively).

1. Complete the definition of `void append(...)` function that receives a pointer to a `CityNode`, and adds that node to the end of the `CityList` (**3 points**).
2. Complete the definition of `void prepend(...)` function that receives a pointer to a `CityNode`, and adds that node to the beginning of the `CityList` (**3 points**).
3. Complete the definition of `CityNode *search(...)` function that receives a city name (a string). It traverses through the elements of the `CityList` to find the city with that name, and returns a pointer to that node if successful. Otherwise, it returns null pointer (**3 points**).
4. Complete the definition of `void printCityList()` function that traverses through the elements of the `CityList`, and calls `printInfo()` on each node's data component (**3 points**).
5. Complete the definition of `void insert(...)` function that receives a pointer to a current `CityNode` along with a pointer to a new `CityNode`. It adds that node to the list as the successor of the current `CityNode`. *Hint:* You must consider special cases where the list is empty, as well as when the current node is the tail of the list (**3 points**).
6. Complete the definition of `void remove(...)` function that receives a pointer to the current `CityNode`, and removes that node (**3 points**).

4 `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 (**3 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 (**3 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