COMP 53: Queues and Deques Lab

Instructions: In this lab, we are going to review the implementation of queues and deques.

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

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

Consider `city.h` and `citynode.h` from the previous lab.Note that class `CityNode` implements doubly-linked nodes, i.e., they support two links: a link to the next node, and a link to the previous node.

2 `citylist.h`

Consider class `CityList` from the previous lab, i.e., doubly-linked list of nodes (without dummy nodes)

```
#ifndef CITYLIST_H
#define CITYLIST_H

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

Complete the definition of five functions above similar to the previous lab ***(5 points)***.

3 `cityqueue.h`

Consider `cityqueue.h` that defines a queue of cities as follows:

```
#ifndef CITYQUEUE_H
#define CITYQUEUE_H

#include "citylist.h"
class CityQueue {
        public:
                CityQueue(CityList &l) { lst = l; }
                void pushCityNode(CityNode *cityNode);
                CityNode *popCityNode();
                CityNode *peekCityNode();
                bool isEmpty();
        private:
                CityList lst;
};
#endif
```

Complete the definition of functions

1. `void pushCityNode(...)` ***(2 points)***
2. `CityNode *popCityNode()` ***(2 points)***
3. `CityNode *peekCityNode()` ***(2 points)***
4. `bool isEmpty()` ***(2 points)***

4 `citydeque.h`

Consider `citydeque.h` that defines a deque of cities as follows:

```
#ifndef CITYDEQUE_H
#define CITYDEQUE_H

#include "citylist.h"
class CityDeque {
        public:
                CityDeque(CityList &l) { lst = l; }
                void pushFrontCityNode(CityNode *cityNode);
                void pushBackCityNode(CityNode *cityNode);
                CityNode *popFrontCityNode();
                CityNode *popBackCityNode();
                CityNode *peekFrontCityNode();
                CityNode *peekBackCityNode();
                bool isEmpty();
        private:
                CityList lst;
};
#endif
```

Complete the definition of functions

1. `void pushFrontCityNode(...)` ***(2 points)***
2. `void pushBackCityNode(...)` ***(2 points)***
3. `CityNode *popFrontCityNode()` ***(2 points)***
4. `CityNode *popBackCityNode()` ***(2 points)***
5. `CityNode *peekFrontCityNode()` ***(2 points)***
6. `CityNode *peekBackCityNode()` ***(2 points)***
7. `bool isEmpty()` ***(2 points)***

5 `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 a `CityList` named as `cityList` ***(1 points)***.
3. Pass `CityList` 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 ***(1 points)***.

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

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

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

(iii) Define a city queue `cityQueue` and initialize it with `cityList` ***(1 points)***.

(iv) Define a city deque `cityDeque` and initialize it with `cityList` ***(1 points)***.

(v) Read the front of the queue and if not null, print out its name and population ***(1 points)***.

(vi) Push Phoenix with the population of 1660472 into `cityQueue`, and then push Santa Fe with the population of 84263 ***(1 points)***.

(vii) Pop the front of the queue ***(1 points)***.

(viii) Read the front of the queue and if not null, print out its name and population ***(1 points)***.

(ix) Read the front of the deque and if not null, print out its name and population ***(1 points)***.

(x) Read the back of the deque and if not null, print out its name and population ***(1 points)***.

(xi) Push Phoenix with the population of 1660472 into the front of the deque ***(1 points)***.

(xii) Push Santa Fe with the population of 84263 into the back of the deque ***(1 points)***.

(xiii) Pop the front of the deque and printing the name and population if it is not null ***(1 points)***.

(xiv) Pop the back of the deque and printing the name and population if it is not null ***(1 points)***.

The output of the program may look like the following:

```
Initializing cityList 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

Reading the front of cityQueue:
Los Angeles: 4340174
Phoenix pushed to cityQueue.
Santa Fe pushed to cityQueue.
Front of cityQueue is popped.
Reading the front of cityQueue:
San Diego: 1591688
Check if cityQueue is empty: 0

Reading the front of cityDeque:
Los Angeles: 4340174
Reading the back of cityDeque:
Eugene: 221452
Phoenix pushed to front of cityDeque.
Santa Fe pushed to back of cityDeque.
Popping the front of cityDeque and printing it:
Phoenix: 1660472
Popping the back of cityDeque and printing it:
Santa Fe: 84263
```