COMP 53: Priority Queues Lab

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

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

1 `city.h`

Consider `city.h` with the following details from the previous lab:

```
#ifndef CITY_H
#define CITY_H

#include<string>

class City {
        public:
                City() {
                        name = "N/A";
                        population = 0;
                }
                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 `cityheap.h`

Consider `cityheap.h` with the following details:

```
#ifndef CITYHEAP_H
#define CITYHEP_H

#include<string>
#include "city.h"
```

```
const int maxArraySize = 100;

class CityMaxHeap {
        public:
                City array[maxArraySize];
                int arraySize;
                CityMaxHeap() {
                        arraySize = 0;
                }
                CityMaxHeap(City arr[], int size) {
                        for(int i = 0; i < size; i++)
                                array[i] = arr[i];
                        arraySize = size;
                        cityHeapify();
                }
                void printHeap();
                void insert(City city);
                void remove();

        private:
                void percolateUp(int nodeInd);
                void percolateDown(int nodeInd, int size);
                void cityHeapify();

};
#endif
```

Class `CityMaxHeap` implements the max heap of cities according to their populations. A `CityMaxHeap` object includes an array of cities along with the size of the array (i.e., heap is stored as an array, rather than a binary tree). Note the two constructors for this class. Complete the definition of missed six functions similar to previous lab ***(6 points)***.

3 cityprioritqueue.h

Consider `cityprioritqueue.h` that defines the priority queue of cities, using the max heap for cities defined in `cityheap.h`:

```
#ifndef CITYPRIORITYQUEUE_H
#define CITYPRIORITYQUEUE_H

#include "cityheap.h"
class CityPriorityQueue {
        public:
                CityPriorityQueue(CityMaxHeap &h) { heap = h; }
                void pushCity(City city);
                void popCity();
                City *peekCity();
                bool isEmpty();
                int getLength();

        private:
                CityMaxHeap heap;
};
#endif
```

Complete the definition of function:

1. `void pushCity(City city)` that inserts `city` after all equal or higher priority (i.e., higher population) cities ***(2 points)***.
2. `void popCity()` that removes the city at the front of the priority queue ***(2 points)***.
3. `City *peekCity()` that returns the address of the city at the front of priority queue (but it does not remove it) ***(2 points)***.
4. `bool isEmpty()` that returns true if priority queue has no items. Otherwise it returns false ***(2 points)***.
5. `int getLength()` that returns the number of cities in the priority queue ***(2 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 (in order):
   (a) Sacramento with population of 505628
   (b) Eugene with the population of 221452
   (c) Stockton with the population of 323761
   (d) Redding with the population of 90292
   (e) San Diego with population of 1591688
   (f) Reno with the population of 289485
   (g) Los Angeles with population of 4340174
   (h) Portland with the population of 730428
   (i) Las Vegas with the population of 711926
   (j) Seattle with the population of 752180
   (k) San Francisco with population of 871421
2. Globally define a `CityMaxHeap` named as `cityHeap` ***(1 points)***.

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

(i) Initialize `cityHeap` according to array `cityArray[]` ***(1 points)***.

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

(iii) Define a city priority queue `cityPQueue` and initialize it with `cityHeap` ***(1 points)***.

(iv) Check if `cityPQueue` is empty ***(1 points)***.

(v) Read the length of `cityPQueue` ***(1 points)***.

(vi) Remove the city at the front of `cityPQueue` ***(1 points)***.

(vii) Read the city at the front of `cityPQueue` ***(1 points)***.

(viii) Insert Phoenix with the population of 1660472 into `cityPQueue` ***(1 points)***.

(ix) Insert Santa Fe with the population of 84263 into `cityPQueue` ***(1 points)***.

(x) Read the city at the front of `cityPQueue` ***(1 points)***.

(xi) Read the length of `cityPQueue` ***(1 points)***.

The output of the program may look like the following:

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

Is cityPQueue empty? 0
Length of cityPQueue: 11

Reading the front of cityPQueue: Los Angeles: 4340174

Removing the item at the front of cityPQueue.
Reading the front of cityPQueue: San Diego: 1591688

Inserting Phoenix with population 1660472 into CityPQueue.
Inserting Santa Fe with population 84263 into CityPQueue.
Reading the front of cityPQueue: Phoenix: 1660472

Length of cityPQueue: 12
```