COMP 53: Heap Lab

Instructions: In this lab, we are going to review (max) heaps and heap sort.

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

1 `city.h`

Consider `city.h` with the following details:

```
#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:
                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();
                void heapSort();

        private:
                City array[maxArraySize];
                int arraySize;
                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.

1. Define function `void printHeap()` that traverses through the array of cities and invokes `printInfo()` on them ***(2 points)***.
2. Define private function `void percolateUp(...)` that receives a node index (in the array). It percolates *up* the node residing in the input index within the heap (by modifying the array of cities) ***(4 points)***.
3. Define private function `void percolateDown(...)` that receives a node index (in the array), along with the size of the array. It percolates *down* the node residing in the input index within the heap (by modifying the array of cities) ***(4 points)***.
4. Define private function `void cityHeapify()` that makes the array of cities a max heap according to their populations. *Hint*: This can be done in a single `for` loop, by invoking `percolateDown(...)` ***(3 points)***.
5. Define function `void insert(...)` that receives a city as input and adds it to the max heap. *Hint*: Use `percolateUp(...)` function. Moreover, do not forget to update the array size ***(3 points)***.
6. Define function `void remove()` that removes the root of the max heap and then adjusts it to satisfy being a max heap. *Hint*: Use `percolateDown(...)` function. Moreover, do not forget to update the array size ***(3 points)***.
7. Define function `void heapSort()` that performs heap sort on the max heap array ***(4 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 (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) Insert city Denver with population 600150 into `cityHeap`. Next, print out the heap ***(1 points)***.

(iv) Remove the root of `cityHeap`. Next, print out the heap ***(1 points)***.

(v) Do heap sort on `cityHeap` and print it out ***(1 points)***.

The output of the program may look like the following:

```
Initializing cityHeap with cityArray[] using appending:
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

Inserting Denver with population 600150 into cityHeap:
Los Angeles: 4340174
San Diego: 1591688
Sacramento: 505628
Portland: 730428
San Francisco: 871421
```

```
Denver: 600150
Stockton: 323761
Redding: 90292
Las Vegas: 711926
Seattle: 752180
Eugene: 221452
Reno: 289485

Removing the root of cityHeap:
San Diego: 1591688
San Francisco: 871421
Sacramento: 505628
Portland: 730428
Seattle: 752180
Denver: 600150
Stockton: 323761
Redding: 90292
Las Vegas: 711926
Reno: 289485
Eugene: 221452

Heap sort of cityHeap:
Redding: 90292
Eugene: 221452
Reno: 289485
Stockton: 323761
Sacramento: 505628
Denver: 600150
Las Vegas: 711926
Portland: 730428
Seattle: 752180
San Francisco: 871421
San Diego: 1591688
```