COMP 53: Search and Sort Lab, part 6

Instructions: In this lab, we are going to review radix 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 **25 points** in aggregate. The details are given in the following.

1 `city.h`

Use `city.h` from the previous lab without any modifications.

2 `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 vector of `City` objects, without initial values. Call it `cityVector` ***(1 points)***.
3. Globally define a vector of unsigned integers, without initial values. Call it `popVector` ***(1 points)***.
4. Pass vectors to these functions as *reference*, and define them as *constant* if the functions are not allowed to modify them.
   (a) Define function `void initVector(...)` that receives a vector of `City` objects, 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 queue with the elements existing in the input array ***(2 points)***.
   (b) Define function `void printCityVector(...)` that receives a vector of `City` objects as input and prints the elements within the vector. *Hint*: You can use range-based `for` loops ***(2 points)***.
   (c) Define function `void printVector(...)` that receives a vector of arbitrary type T as input and prints the elements within the vector. *Hint*: You can use range-based `for` loops ***(2 points)***.

(d) Define function `void populationProject(...)` that receives a vector of `City` objects as input, along with a vector of unsigned integers. It inserts the population of each city in the first vector to the second vector. *Hint*: You can use range-based `for` loops ***(2 points)***.

(e) Define function `int radixGetLength(...)` that receives an unsigned integer as input and returns the number of digits of that input value ***(2 points)***.

(f) Define function `int radixGetMaxLength(...)` that receives a vector of unsigned integers and returns the maximum length (i.e., number of digits) among the values within the input vector ***(2 points)***.

(g) Define function `void populationRadixSort(...)` that receives a vector of unsigned integers as input. It does radix sort on that vector (by invoking the `radixGetMaxLength()` function on sorted vectors. *Hint*: You can implement a bucket by a list, i.e., an array of lists (`list<unsigned int>` bucket[10]) ***(6 points)***.

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

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

(ii) Print out the entries of `cityVector`, using the appropriate function defined above ***(1 points)***.

(iii) Initialize `popVector` according to `cityVector[]` using the function defined above ***(1 points)***.

(iv) Print out the entries of `popVector`, using the appropriate function defined above ***(1 points)***.

(v) Do radix sort on `popVector` and print out the updated vector ***(1 points)***.

The output of the program may look like the following:

```
Initializing cityVector with cityArray[]:
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


Initialize popVector according to cityVector:
4340174 1591688 871421 505628 323761 90292 711926 289485 730428 752180 221452

Radix sort on popVector:
90292 221452 289485 323761 505628 711926 730428 752180 871421 1591688 4340174
```