Skills Assessment 3

Description:
In this skills assessment, you will implement two functions **intersectionOfTwoArrays** and **ropesGameCost**

starter.zip (Starter Code):
The following is a brief description of the starter code provided to you:

- **ques.cpp**
  - Contains the non-functional definitions of the *intersectionOfTwoArrays* and *ropesGameCost* functions
- **PriorityQueueHeap.h**
  - You may use the methods provided in this class if you wish to use a priority queue data structure for any of your implementations. The method declarations in the PriorityQueueHeap class are the same as in PA3. This PriorityQueueHeap class implements a min heap.

Task
You must implement the **intersectionOfTwoArrays** and **ropesGameCost** functions.

Question 1 (25 points)
intersectionOfTwoArrays Function

- The ***vector<int> intersectionOfTwoArrays(vector<int> nums, vector<int> otherNums)*** function takes in two vectors of integers as parameters.
- **Given these two integer vectors, return *a vector of their intersection*.** Each element in the result must appear as many times as it shows in both vectors and you may return the result in any order.
  - In other words, if an element appears in both vectors, the number of times this element must appear in the resulting intersection is **minimum(number of times element appears in *nums*, number of times element appears in *otherNums*)**
  - **If there is no element common to both the vectors, return an empty vector.**
- **Note that you must implement this function in O(N + M) time where N is the size of nums and M is the size of otherNums, else you will not get any points for this question.**
  - You can use the vector's **.size()** method to get the size of the vector; you can use the **.push_back()** method to add an element to the end of a vector.

- **Example:**

Input 1:
vector<int> nums = [9, 8, 8, 7]
vector<int> otherNums = [8, 8, 8, 9]

intersectionOfTwoArrays(nums, otherNums)
Output: [8, 8, 9]
Reason: Both nums and otherNums have two 8's and one 9

Input 2:
vector<int> nums = [1, 8, 1, 8, 1, 7, 46, 1]
vector<int> otherNums = [8, 1, 8, 1, 1, 203]

intersectionOfTwoArrays(nums, otherNums)
Output: [8, 8, 1, 1, 1]
Reason: Both nums and otherNums have two 8's and three 1's
Note: The order of the common numbers do not matter, you may return them in any order. For example, another acceptable solution is [1, 8, 1, 8, 1].

Input 3:
vector<int> nums = [1, 8, 7, 5, 53]
vector<int> otherNums = [102, 203, 201, 103, 106, 107]

intersectionOfTwoArrays(nums, otherNums)
Output: []
Reason: No element is common to both nums and otherNums

Question 2 (25 points)
ropesGameCost Function

- The ***int ropesGameCost(vector<int> ropes)*** function takes in a single vector of integers representing the length of ropes as a parameter.
  - ropes[i] is the length of the rope at position i.
- We are playing a game with ropes. On each turn, we choose the **shortest two ropes** and create one combined rope. Suppose the shortest two ropes have weights x and y with x <= y. The result of combining them would be one rope of length x + y and we incur a cost of x + y by combining them.
  - **At the end of the game, there is one rope left. You must return the total cost (i.e. cumulative cost) associated with playing this game.**

- **Note that you must implement this function in O(N log N) time where N is the size of the "ropes" vector, else you will not get any points for this question.**

- **Example:**

Input 1:
vector<int> ropes = [3, 8, 4, 6, 7]

ropesGameCost(ropes)
Output: 63
Reason:
1) First, connect two ropes of length 3 and 4. Cost of connecting these ropes is 3 + 4 = **7.** Now we have ropes of length { 8, 6, 7, 7 }
2) Then, connect the ropes with lengths 6 and 7. Now the cost of connecting these ropes is 6 + 7 = **13.** Now we have ropes of length { 8, 7, 13 }
3) Then, connect the ropes with lengths 7 and 8. Now the cost of connecting these ropes is 7 + 8 = **15.** Now we have ropes of length { 13, 15 }
3) Finally, connect the two last ropes with cost 13 + 15 = **28.**

So, Total Cost of connecting these ropes in the above order is 7 + 13 + 15 + 28 = 63.

Input 2:
vector<int> ropes = [3, 7, 9, 4]

ropesGameCost(ropes)
Output: 44
Reason:
1) First, connect the ropes of length 3 and 4. Cost of connecting these ropes is 3 + 4 = **7.** Now we have ropes of length { 7, 7, 9 }
2) Then, connect the ropes with length 7 and 7. Now the cost of connecting these ropes is 7 + 7 = **14.** Now we have ropes of length { 14, 9 }
3) Finally, connect the two last ropes with cost 9 + 14 = **23.**

So, Total Cost of connecting these ropes in the above order is 7 + 14 + 23 = 44.

You have been provided with these sample test cases in ques.cpp that you can run on both your functions to verify that these sample test cases are working correctly on both of your functions.

- **Note: If you wish to create helper functions, please create the helper functions within the respective header guards of each question. This will allow us to isolate the two questions from each other when autograding. Please do not make any changes to the header guards given to you in the starter file.**

Also note that the sample test cases provided are a good indication that your logic is correct but they may not necessarily reflect your final score on the hidden autograded test cases. You are encouraged to create additional sample test cases in main() to test your code.

Deliverables and Submissions:

In a zip folder named using the format **<FirstName>-<LastName>-<UIN>-<SA3>.zip** you must submit the following file to Canvas:

- **ques.cpp (**containing your functional implementation of the intersectionOfTwoArrays and ropesGameCost methods)

Do not use angular brackets in the name of the submission folder.