Gurveen Nanua, gkn4, 301546459

Design Documentation: HERE

- I created a class called IPQ that uses template type T and an initial queue capacity size. (with size = 10)
- I started my index for the array at 0 instead of 1 because I found it easier for me to understand it. (I'm more used to this)

Private:

- In my IPQ class I made two private internal classes:
    1) **HeapPair -** in this class I initialized the ID and priority. I then created a constructor with the ID/priority for the array to use.
    2) **minHeap -** in this class I initialize a pointer to HeapPair, size and capacity. I made a constructor that creates a new array heap with elements of HeapPair, I set the capacity and size to 0.

  In this class I made the following operations:
- per_up(int index) **-** percolates up from the given index
- per_down(int index) - percolates down from the given index
- reserve(int newCapacity) - called from the public reserve function and creates/copies a new array given the elements from the previous array with a bigger capacity
- swap(int indx1, int indx2) - just swaps 2 elements in the list
- And there's a destructor

- I decided to make two separate classes because I found it easier to implement a new array that contains both ID and priority at each index.
- In addition, my private contains a pointer of minHeap *heap and an unordered_map called map that takes <string, int> to keep track of indices of each ID.

Public:

- I made two constructors. One makes an empty heap and the other gets two vectors, (one vector of strings and the other ints) and makes a heap. To make it simpler for the vector constructor, I used the insert function already in the class to make the heap correctly.
- I had some difficulty with my second constructor because I was unsure how to initialize a vector, but I figured it out in the end.
- I then made the other functions required in the assignment, using the heap-> pointer to call functions from the minHeap such as per_up/per_down to maintain the order. Also to update the size and check capacity.
- While inserting I made sure to update the map so that each ID was in the correct index while I inserted or removed elements.
- For update priority, it changes the priority by finding the index of the key using the map function map.at(ID).

- My reserve function calls heap->reserve(i) from the minHeap class where it then reserves a new capacity if the capacity given is greater than the one that already exists.

Testing Regimen: testIPQ.cpp