

Assignment 8: Find  $k^{th}$  largest number of an unsorted list

## Description

Given a set of  $n$  integers, find the  $k - th$  largest element in the list. You will need to create the following class which could be either a min heap or a max heap.

```
class priorityQ
{
public:
    priorityQ(int = 10);
    priorityQ(const priorityQ<Type>&);
    ~priorityQ();
    const priorityQ<Type>& operator=(const priorityQ<Type>&);
    void insert(const Type&);
    void deleteHighestPriority();
    Type getHighestPriority() const;
    bool isEmpty() const;
    void bubbleUp(int);
    void bubbleDown(int);
    int getSize() const;
private:
    int capacity;
    int items;
    Type * heapArray;
};
```

- `Type * heapArray` - array that contains the elements for priority queue
- `int capacity` - the length of the heapArray
- `int items` - the amount of items currently stored in the heap
- `priorityQ<Type>::priorityQ(int capacity)` - sets `this->capacity` with `capacity`, allocates a dynamic array to `heapArray`, and sets `items` to 1 or 0
- `priorityQ<Type>::priorityQ(const priorityQ<Type>& copy)` - copy constructor, performs a deep copy of the copy object to the `*this` object
- `priorityQ<Type>::~~priorityQ()` - destructor

- `const priorityQ<Type>& priorityQ<Type>::operator=(const priorityQ<Type>& rhs)` - assignment operator, performs a deep copy of `rhs` object into `*this` object, remember to check for a self assignment and deallocate `this->heapArray` first before performing the actual deep copy
- `void priorityQ<Type>::insert(const Type& element)` - inserts element to the end of the heap and bubbles the element up, resizes if needed, also increments items counter by 1
- `void priorityQ<Type>::deleteHighestPriority()` - removes the root element by assigning the root with the end element in the heap and bubbles the element down, also decrements items by 1 (does nothing if the heap is empty)
- `Type priorityQ<Type>::getHighestPriority() const` - returns the highest priority item, the item at index 1 of the `heapArray`
- `bool priorityQ<Type>::isEmpty() const` - returns `true` if there are no items in the heap and `false` otherwise
- `void priorityQ<Type>::bubbleUp(int index)` - bubbles up an element in `heapArray` at index "index", and keeps bubbling up as needed, remember the parent of element `x` in the `heapArray` is at index  $x / 2$
- `void priorityQ<Type>::bubbleDown(int index)` - bubbles down an element from index "index", and keeps bubbling down as needed, remember the left child and right child of element `x` is at location  $2 * x$  and  $2 * x + 1$  respectively, you always bubble down with the highest priority child, also remember if  $2 * x > \text{items}$  then `x` is a leaf node and no bubbling down is needed
- `int priorityQ<Type>::getSize() const` - returns the amount of elements stored in the heap

## Contents of main

You will prompt for an input file, the first line contains an integer  $K$  that denotes the  $K - th$  largest number you want to find. Then a list of integers (one integer per line) and each line is ended with an end of line character. You need to output the  $k - th$  largest element whenever a new integer read in causes an update (a new  $k - th$  largest number is found).

You need to use a `priorityQ<int>` object to make this work, you can not sort the array or insert the entire list into a `priorityQ<int>` object and then just pop the heap  $X$  number of times. You have to assume that the entire sequence of numbers is not known in advance.

## Example Output

```
$ g++ main.cpp
$ ./a.out
Enter filename: input01.txt
Initial 12-th largest number: 88
New 12-th largest number: 86
New 12-th largest number: 82
New 12-th largest number: 79
New 12-th largest number: 78
New 12-th largest number: 65
New 12-th largest number: 62
New 12-th largest number: 59
New 12-th largest number: 57
New 12-th largest number: 47
New 12-th largest number: 44
New 12-th largest number: 43
```

```

New 12-th largest number: 39
New 12-th largest number: 34
New 12-th largest number: 30
New 12-th largest number: 29
New 12-th largest number: 27
New 12-th largest number: 26
New 12-th largest number: 24
New 12-th largest number: 23
Final 12-th largest number: 23

$ ./a.out
Enter filename: input02.txt
Initial 27-th largest number: 99
New 27-th largest number: 90
New 27-th largest number: 88
New 27-th largest number: 86
New 27-th largest number: 82
New 27-th largest number: 79
New 27-th largest number: 78
New 27-th largest number: 76
New 27-th largest number: 73
New 27-th largest number: 70
New 27-th largest number: 69
New 27-th largest number: 66
New 27-th largest number: 65
New 27-th largest number: 63
New 27-th largest number: 62
New 27-th largest number: 61
New 27-th largest number: 59
New 27-th largest number: 57
New 27-th largest number: 56
New 27-th largest number: 54
New 27-th largest number: 48
New 27-th largest number: 47
Final 27-th largest number: 47

$ ./a.out
Enter filename: input03.txt
Initial 45-th largest number: 99
New 45-th largest number: 96
New 45-th largest number: 90
New 45-th largest number: 88
New 45-th largest number: 86
New 45-th largest number: 82
Final 45-th largest number: 82

```

## Specifications

- Comment your code and your functions
- Make sure your code is memory leak free
- Do not sort the list of just insert every element from the file into the priority queue at once

## Submission

Upload your source code and write up to the class site by the deadline, submit `main.cpp` and `priorityQ.hpp` to code grade

## References

- The image used in this write can be found at <https://www.deviantart.com/solarstormtm/art/Rocko-from-Rocko-s-Modern-Life-395651928>