Lab 4: Priority Queues and Heaps with AlgoFy

You are working for a promising new music streaming service called "AlgoFy." AlgoFy would like to offer a new ranking feature that will track the top *k* most streamed songs of the day.

You have been tasked with building a class, "AlgoFy" that can track the most streamed songs and return an accurate list top *k* list at any point in time.

Another team has already developed the song streaming feature, "iStream," and you will need to integrate with it. iStream will send AlgoFy play data as it processes customer requests. This data will be batched so the AlgoFy class will need to ingest lists of played songs multiple times.

A song is identified with an integer *id* number.

The AlgoFy class will need the following methods:

- constructor(k): the number of songs to track in the top k
- streamSongs(list of songIds): receive a batch of streamed songs => updates internal state
- getTopK(): return the top k most streamed songs in order

As an example, the AlgoFy class may see the following usage:

```
ranker = new AlgoFy(2)
ranker.streamSongs([1, 1, 2, 2, 3])
ranker.streamSongs([3, 3, 3, 1, 2, 1, 3])
ranker.getTopK() => [3, 1]
ranker.streamSongs([2, 2, 2, 2, 2])
ranker.getTopK() => [2, 3]
```

You must solve this problem by effectively using the properties of a Priority Queue or Heap.

Deliverable Explanation

You must efficiently solve this problem in either Python or Java, using one of the attached starter code files. Several unit tests are provided to help you test your code.

Having the optimal runtime will grant you extra credit. On the other hand, you may also lose points if your code is substantially inefficient. You must also provide an explanation of the running time of your code in the provided placeholder in the starter code.

Your code will be evaluated against a set of visible and hidden test cases (the visible ones are also provided in the starter code for your convenience) to test correctness. The hidden test cases are intended to verify that you have a robust solution that considers possible edge cases and various inputs.

If you are using Java, your file's package must be set to "student".

Submission Instructions

Submit the following file on Gradescope:
AlgoFy.java to Lab4 (Java)
OR
algo_fy.py to Lab4 (Python)

Grading Methodology

The lab is worth 100 points and is broken down into the following categories:

- Test Cases Passing (80 points)
- Big O explanation (15 points)
- Overall code performance and style (5 points)
- Having Optimal Running Time (5 points of Extra Credit)