PRIORITY QUEUE

ASPECT INC.

1. Description of Service

Tim is an IT guy. He executes work orders by people randomly coming to his desk and asking for things. Executives have been complaining to Tim's manager that Tim isn't doing the most important things in the right order and is getting interrupted too often. Tim's manager has had enough of this and has initiated a program in the organization to fix it. The engineering group is commissioned with writing a web service that accepts work orders and provides Tim with a prioritized list to work. Other groups in the company have been asked to interface with this web service and submit their work orders to it.

You are the engineer that has been given the job of writing this priority queue web service (you are not being asked to write the systems that submit the work orders, remember). For this first iteration of the product, the work order text itself is not included in the web service request. Tim will be expected to go talk to the requestor when it is time for him to work the order. Do a good job; make Tim and his manager happy campers.

Your manager has given you the following constraints about how this web service should be implemented:

- The service should implement a RESTful interface.
- The service will accept work order requests from other services and place them into a queue.
- A work order request is comprised of the ID of the requestor and a Date.
- The service should reject work orders with IDs that already exist in the queue (I.E. the same user cannot have more than one work order in the queue).
- The IDs will be integers in the range of 1 to 9223372036854775807.
- There are 4 classes of IDs, normal, priority, VIP, and management override.
- You can determine the class of the ID using the following method:
  (1) IDs that are evenly divisible by 3 are priority IDs.
  (2) IDs that are evenly divisible by 5 are VIP IDs.
  (3) IDs that are evenly divisible by both 3 and 5 are management override IDs.
  (4) IDs that are not evenly divisible by 3 or 5 are normal IDs.
- IDs must be sorted in the queue according to a formula that varies based on their class.

(1) Normal IDs are given a rank equal to the number of seconds they've been in the queue.
(2) Priority IDs are given a rank equal to the result of applying the following formula to the number of seconds they've been in the queue:

$$\max(3, n \log n)$$

(3) VIP IDs are given a rank equal to the result of applying the following formula to the number of seconds they've been in the queue:

$$\max(4, 2n \log n)$$

(4) Management Override IDs are always ranked ahead of all other IDs and are ranked among themselves according to the number of seconds they've been in the queue.

- The queue should be sorted from highest ranked to lowest ranked ID.
- The format of times passed to the services will be defined by you and should be noted somewhere.

2. Description of Endpoints

You should provide the following endpoints:

(1) An endpoint for adding a ID to queue (enqueue). This endpoint should accept two parameters, the ID to enqueue and the time at which the ID was added to the queue.
(2) An endpoint for getting the top ID from the queue and removing it (dequeue). This endpoint should return the highest ranked ID and the time it was entered into the queue.
(3) An endpoint for getting the list of IDs in the queue. This endpoint should return a list of IDs sorted from highest ranked to lowest.
(4) An endpoint for removing a specific ID from the queue. This endpoint should accept a single parameter, the ID to remove.
(5) An endpoint to get the position of a specific ID in the queue. This endpoint should accept one parameter, the ID to get the position of. It should return the position of the ID in the queue indexed from 0.
(6) An endpoint to get the average wait time. This endpoint should accept a single parameter, the current time, and should return the average (mean) number of seconds that each ID has been waiting in the queue.

Endpoints should follow REST best practices, and parameters should be passed in the fashions that most closely match with REST principles.