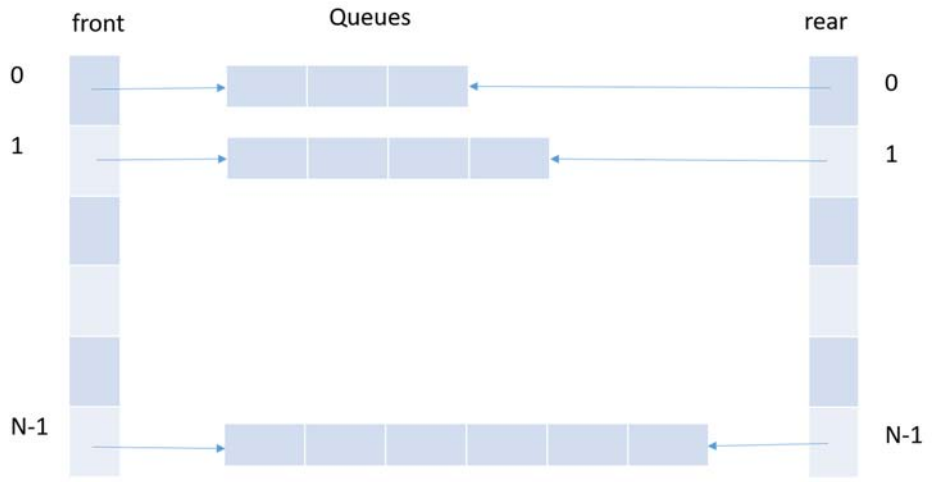
Problem 2: Queue

We aim to implement a quality of service system in a network router so that it can differentiate the type of traffic traversing it, and it can give privilege to data belonging to a given traffic compared to other type of data. So we will create N queues based on *linked list* each. You have an array of pointers to the front of queues and an array of pointers to the rear of the queues, as shown in the figure. Each node of the queues contains the *type* and *data*, and the necessary pointers for connecting with other nodes. Queue qi contains the data (and type) of traffic of type i, such that type i has a higher priority than j, if i<j. It means when the data arrives to the system, they are stored in the necessary queue according to its type. The data leaves the system according to their priority, from the higher priority to the lower one; priority i before j, if i<j. It means, q0 dequeue all its data before q1, then q1 dequeue all its data before q2, etc. qj does not dequeue its data before qi, if i<j.



To simplify the task, the data arriving to the system are all 0. You should generate a random number ni for each qi that corresponds to the number of data arriving to the queue qi with type i. So you receive n_1, n_2, …, n_{N-1} data then you serve (output from the system) n random data (starting from the queue with high priority, it means lower type i). Repeat this reception and service many times and calculate the size of each queue and then the average size of each queue.

- Write a C++ program that solves this problem. What should be the relation between ni for all i and n so that the system will not overload?

Example: N=3; q1, q2 and q3 initially empty

Sample Input				**Sample Output**		
n1	**n2**	**n3**	**n**	**q1.size()**	**q2.size()**	**q3.size()**
3	4	5	6	0	1	5
2	3	1	8	0	0	4
4	5	3	5	0	4	7
				Average		
				0	1.67	5.33