Implement K Queues In Single Array [CodeStudio](https://www.codingninjas.com/studio/problems/n-queue-using-array_1170053?leftPanelTab=0)

This C++ program demonstrates the implementation of multiple queues (K queues) in a single array using a custom class **KQueues**. The class provides various methods to enqueue, dequeue, and manage each queue within the array.

1. **KQueues Class**: This class is designed to manage multiple queues within a single array.
   * **arr**: An array to store elements.
   * **front**: An array to store the front indices of each queue.
   * **rear**: An array to store the rear indices of each queue.
   * **next**: An array to maintain pointers to the next elements in the array.
   * **freespot**: An index to keep track of the next free spot in the array.
2. **Constructor (KQueues(int k, int n))**: Initializes the arrays and pointers. **k** represents the number of queues, and **n** represents the total capacity of the array.
3. **enQueue(int x, int m)**: Enqueues an element **x** into queue **m**. It checks if there is space available in the array and updates front and rear pointers accordingly.
   * **Time Complexity: O(1)**
   * **Space Complexity: O(1)**
4. **deQueue(int m)**: Dequeues an element from queue **m**. It checks if the queue is empty and updates front and rear pointers. The removed element is marked as a free spot.
   * **Time Complexity: O(1)**
   * **Space Complexity: O(1)**
5. **isEmpty(int m)**: Checks if queue **m** is empty by examining the front pointer.
   * **Time Complexity: O(1)**
   * **Space Complexity: O(1)**
6. **isFull(int m)**: Checks if queue **m** is full by examining the **freespot** pointer.
   * **Time Complexity: O(1)**
   * **Space Complexity: O(1)**
7. **getFront(int m)**: Returns the index of the front element in queue **m**.
   * **Time Complexity: O(1)**
   * **Space Complexity: O(1)**
8. **getRear(int m)**: Returns the index of the rear element in queue **m**.
   * **Time Complexity: O(1)**
   * **Space Complexity: O(1)**
9. **getSize(int m)**: Returns the size (number of elements) in queue **m**.
   * **Time Complexity: O(n) in the worst case, where n is the total number of elements in all queues.**
   * **Space Complexity: O(1)**