Problem 4: Implementation of Queues

Data Structures Lab (CS111)

In this practice problem, you need to implement queues using different approaches. This problem has two parts:

1. Create a queue with array of size 5. Then implement following operations for the created queue:

   - *enqueue*(), *dequeue*(), *queuesize*(). **[5 marks]**

2. You need to implement *multQueue* according to the following instructions. Implement a circular queue called *waitQueue* using an integer array of length 10. Initially that queue is empty. Next implement a linked list called *serviceList* with initially one node where the node contains a circular queue implemented using array of size 5 where we can insert integer elements. Now you need to implement operations to support following operations:

   (a) *multEnqueue(int x)*: If the *waitQueue* is empty, It should enqueue x to the queue in the *serviceList*. If the queue in the *serviceList* is full, then it should insert the element into the *waitQueue*. If *waitQueue* is also full, then it should create a new node in the *serviceList* with a circular queue of length 5. Then it should dequeue five elements from *waitQueue* and enqueue those elements into the circular queue of newly created node in the *serviceList*. After that it should enqueue x to the *waitQueue*.

   (b) *multDequeue()*: In this operation, you should dequeue an element from the first node of the list *serviceList* and return it. Then, dequeue one element from *waitQueue* and enqueue that to the queue from where that element is dequeued in the *serviceList*. If the *waitQueue* is empty and due to this dequeue operation the queue at the first node of the *serviceList* is also empty, then delete that node if it is not the only node in *serviceList*. Note that, when first node gets deleted, next node of the first node becomes the first node.

   (c) *statusQueue()*: This should print the number of nodes in the *serviceList*, elements in the queues of each node and elements in the *waitQueue*. It should also show that if a new element comes, then where it will be placed in the current *multQueue*. If you cannot place it without creating a new node, you should return that *multQueue* is full.

   You can implement and use other supporting functions also. **[15 marks]**