I suggested the following extension to the Gibbons D algorithm in order to support the *deq(⟘)* operation. This will require us to add a step to the algorithm (denoted as step 4’) as well as make a slight change to step 4 to avoid early termination. The changes are presented as follows:

Given an input of *N* operations of the type *enq/deq*, *K < N* of which are of the form *deq(⟘)* adjust step 4 of Gibbons’ algorithm to ignore all *deq(⟘)* operations. As these operations are not matched this will avoid an early termination on step 4. Note that all standard limitations still apply on the values. Specifically, that they are still distinct integer pairs. In addition, step 4’ is added between steps 4 and 5 in which we add the pairs to according to the following rule: .   
In principle, we are making sure that *deq(⟘)* operations never appear between the *enq* and *deq* operations of any single value.

Theorem:

The revised algorithm’s correctness is still sound. Specifically, if the algorithm approves a trace it means that all *deq(⟘)* operations in it behave correctly.   
Proof for *deq(⟘)*: for a *deq(⟘)* operation to be invalid, it means that it occurs when a queue is not empty. Therefore, the following statement should be correct: . However, since are an event pair, this is in direct contradiction with the pairs added in step 4’. Moreover, the transitive pairs added in step 5 make sure that this applies globally and iteratively throughout the algorithm’s entire run.  
The full algorithm’s correctness still remains sound as we only added pairs and never removed pairs from the original algorithm’s run.