

<<Start Of Thread 1>>

```

1  [TestMethod]
2  [HostType("Chess")]
3  [TestProperty("ChessMonitorVolatiles","True")]
4  public void TestLockFreeQueueWithError1(){
5      QueueTest qt = new QueueTest();
6      LockFreeQueue<String> lfq =
7          new LockFreeQueueWithError1<String>();
8      EnqContainer firstParams, secondParams;
9      public struct EnqContainer {
10         public String first;
11         public String second;
12         public LockFreeQueue<String> q;
13     }
14     firstParams.q = secondParams.q = lfq;
15     firstParams.first = "a";
16     firstParams.second = "b";
17     secondParams.first = "c";
18     secondParams.second = "d";
19     Thread t2 =
20         new Thread(new ParameterizedThreadStart(
21             qt.EnqueueTwoStrings));
22     t2.Start(secondParams);
23     EnqTwoStrings(firstParams);
24
25
26     public void EnqTwoStrings (Object o) {
27         EnqContainer e = (EnqContainer)o;
28         e.q.Enqueue(e.first);
29         public override void Enq(T x) {
30             <x == "a">
31             Node<T> node = new Node<T>(x);
32             <1st Iteration of 1>
33             while (true) {
34                 Node<T> last = tail;
35                 Node<T> next = last.next;
36                 <True>
37                 if (last == tail) {
38                     <True>
39                     if (next == null) {
40                         <True>
41                         if (next == Interlocked.CompareExchange(
42                             ref last.next,node,next)) {
43                             // The mistake was introduced by
44                             // replacing an atomic operation
45                             // (i.e. CompareExchange) with a
46                             // non-atomic operation.
47                             if (tail == last)
48                                 tail = node;
49                             return;
50                         }
51                     }
52                     <Does Not Execute>
53                     else {
54                         ...
55                     }
56                 }

```

KEY



Enter into a function call



Shows the order of execution.

<<Start Of Thread 2>>

```

public void EnqTwoStrings (Object o) {
    EnqContainer e = (EnqContainer)o;
    e.q.Enqueue(e.first);
    public override void Enq(T x) {
        <x == "c">
        Node<T> node = new Node<T>(x);
        <1st Iteration of 2>
        while (true) {
            Node<T> last = tail;
            Node<T> next = last.next;
            <True>
            if (last == tail) {
                <False – Does not Execute>
                if (next == null) {
                    ...
                }
                <Does Execute>
            } else {
                Interlocked.CompareExchange(
                    ref tail, next, last);
            }
        }
        <2nd Iteration of 2>
        while (true) {
            Node<T> last = tail;
            Node<T> next = last.next;
            <True>
            if (last == tail) {
                <True>
                if (next == null) {
                    <True>
                    if (next == Interlocked.CompareExchange(

```

