53,用反证法。假设存在针对线程A，B，C的一致性协议，则存在一个临界状态，是的A的下一个迁移pop()到达0价状态s0，而B的下一个迁移到达1价状态s1 此时C从状态s0开始运行会决定0，但从s1开始运行会决定1，但对C来说状态s0和s1不可分，因此矛盾，Stack的一致数比3小。通过类比可得双线程一致性代码。

```
public class StackConsensus<T>{
    private static final int WIN = 0;
    private static final int LOSE = 1;
    Stack s;
    public StackConsensus(){
        s = new Stack();
        s.push(WIN);
        s.push(LOSE);
    }

    public T decide(T value){
        propose(value);
        int status = s.pop();
        int i = ThreadID.get();
        if (status == WIN)
            return proposed[i];
        else
            return proposed[1-i];
    }
}
```

54,如代码所示，该协议是无等待的，并且每个线程都返回第一个压入队列的线程数，则一致数是无限的。

```java
public class QueueConsensus<T>{
    Queue q;
    public QueueConsensus(){
        q = new Queue();
    }

    public T decide(T value){
        propose(value);
        q.enq(ThreadID.get());
        if(ThreadID.get() == q.peek())
            return proposed[ThreadID.get()];
        else
            return proposed[q.peek()];
    }
}
```

68,第二三步有问题第二步错误：因为peek()可以实现一致数为n的协议，但是快照对象一致数为1，所以无法用其构造采用peek构造的一致数为n的协议。第三步错误：所构造的协议不保证一致性，如A.deq()操作中第11行int limit = head.get()，操作结束时，B.deq()可以完成该行操作，此时二者同时进入items数组查找数据，则A，B返回的数据将不符合运行顺序。

78,无锁算法不会出错。如果只改变tail的序号为1的话，无等待算法会出错，第12行 Node help = announce[(before.seq+1)但如果改变一下将Node初始时的seq和判断是否被添加进链表里是设为-1就不会有问题

80,该方法是可行的，如果有一个线程A一直处在不被一致性算法选中的状态，根据(before.seq + 1)