提案（1.1, *Y*）。

（6）S1 发送 Accept（3.1, *Y*）消息至 S1 和 S3，该提案被成功接受。S1 批准了提案值 *Y*。

在这种情况下，批准了两个不通过的提案值，违反了 Basic Paxos 算法的要求。

可见，接受者收到 Accept 消息后必须更新承诺编号，如果第（3）步 A3 将承诺提案编号更新为 2，那么第（4）步的提案将不被接受，这样算法才正确。

虽说 Paxos 两阶段可以发送给不同的多数派，但一定要注意仔细处理状态更新，否则算法将不能正确达成共识。

注： *以下试题来自 Diego Ongaro 和 John Ousterhout。*

5. 假设一个提议者以提案值 v1 运行 Basic Paxos，但它在算法执行过程中或执行后的某个（未知）时间点宕机了。假设该提议者重新启动并从头开始运行协议，使用相同的提案编号但不同的提案值 v2 来提出提案，这样安全吗？请解释你的答案。

参考答案：不安全。不同的提案必须具有不同的提案编号。下面是一个 3 节点集群的例子：

（1）S1 发送 `Prepare(n=1.1)`消息至 S1 和 S2。

（2）S1 发送 `Accept(n=1.1, v=v1)`消息至 S1 就宕机了。

（3）S1 重启。

（4）S1 发送 `Prepare(n=1.1)`消息至 S2 和 S3，并且发现没有任何节点返回被接受的提案。

（5）S1 发送 `Accept(n=1.1, v=v2)`消息至 S2 和 S3。

（6）S1 将 v2 被批准的消息返回给客户端。

（7）S2 收到新的客户端请求，发送 `Prepare(n=2.2)`消息至 S1 和 S2，并收到来自 S1 的响应（acceptedProposal=1.1, acceptedValue=v1）和来自 S2 的响应（acceptedProposal=1.1, acceptedValue=v2）。

（8）S2 直接选择 v1 作为提案值。

（9）S2 发送 `Accept(n=2.2, v=v1)`至 S1、S2 和 S3。

（10）S2 将 v1 被批准的消息返回给客户端。

在这种情况下，系统批准了两个不同的提案值，显然违反了 Basic Paxos 算法的要求。

可能出现的另一个问题是：

（1）S1 发送 `Prepare(n=1.1)`消息至 S1 和 S2。

（2）S1 发送 `Accept(n=1.1, v=v1)`消息至 S1，收到成功响应。

（3）S1 发送 `Accept(n=1.1, v=v1)`消息至 S2 和 S3，但也许因为网络分区，它们并没有收到请求。

（4）S1 重启。