Distributed Algorithms CPSC-561
Assignment 2

Michael McCulloch

1 LL/SC

For all of the following algorithms, they are each wait-free, as they do not contain a loop.

A) CAS from SCAS

Algorithm 1 CAS from SCAS

```
function $CAS_p$(old,new)
    old' = SCAS.scas(old, new)
    return old' = old
end function
```

Let the invocation of SCAS.scas() be the linearization point X. *Claim:* If a process P executes X before another process Q executes X, then the invocation of P is ordered before the invocation of Q. *Proof:* By the definition of SCAS, if it returns the same value as it's first argument, it must have succeeded. If this value is different, another process Q must have finished it's invocation between the invocation of SCAS and the assignment of it's result to old' within P, a contradiction.

B) SCAS from CAS

Algorithm 2 SCAS from CAS

```
function $SCAS_p$(old,new)
    success = CAS.cas(old, new)
    if success then
        return old
    else
        return CAS.read()
    end if
end function
```

Let the invocation of CAS.cas() be the linearization point X. *Claim: see above. Proof:* If CAS is successful, then the old value passed in must be the old value. If CAS is unsuccessful, then a different value must have been the old value, which can only have been cause if another process executed X first. A contradiction.

C) CAS from LL/SC

Algorithm 3 CAS from LL/SC

```
1: function $CAS_p$(old, new)
2:     old' = LL/SC.LL()
3:     if old' $\neq$ old then
4:         return false
5:     else
6:         return LL/SC.SC(new)
7:     end if
8: end function
```

Claim: If a process P executes lines two through six, before another process Q executes the same, then P's invocation is ordered before Q's. *Proof:* If P returns `false`, then another process must have executed either LL() or SC() successfully....?

2 Consensus

3 SRSW