Option Exercises (Chapter 7, numbers 7.6 and 7.7 in book)

6 Based on Dekker/Dijkstra algorithm:

Two warring neighbors are separated by a field with wild berries. They agree to permit each other to enter the field to pick berries, but also need to ensure that only one of them is ever in the field at a time. After negotiation, they agree to the following protocol:

When one neighbor wants to enter the field, he raises a flag. If he sees his neighbor's flag, he does not enter but lowers his flag and tries again. If he does not see his neighbor's flag, he enters the field and picks berries. He lowers his flag after leaving the field.

Model this algorithm for two neighbors `n1` and `n2`. Specify the required *safety* property for the field and check that it does indeed ensure mutually exclusive access. Specify the required *progress* properties for the neighbors such that they both get to pick berries given a fair scheduling strategy. Are there any adverse circumstances in which neighbors would not make progress? What if the neighbors are greedy?

Hint: The following FSP can be used to model the flags:

```
const True = 1        const False = 0
range Bool = False..True
set BoolActions =
     {setTrue,setFalse,[False],[True]}

BOOLVAR = VAL[False],
VAL[v:Bool] = ( setTrue -> VAL[True]
              | setFalse -> VAL[False]
              | [v] -> VAL[v]
              ).

||FLAGS = (flag1:BOOLVAR||flag2:BOOLVAR).
```

7. Peterson's Algorithm for two processes (Peterson G.L. 1981):

Fortunately for the neighbors in question 7.6, Gary Peterson visits one day and explains his algorithm to them. He explains that, in addition to the flags, the neighbors must share a *turn* indicator which can take value 1 or 2. This is used to avoid potential deadlock. The algorithm is as follows:

When one neighbor wants to enter the field, he raises his flag and sets turn to indicate his neighbor. If he sees his neighbor's flag and it is his neighbor's turn, he may not enter but must try again later. Otherwise, he can enter the field and pick berries and must lower his flag after leaving the field.

For instance, neighbor n1 behaves as shown below, and neighbor n2 behaves symmetrically.

```
While (true) {
      flag1 = true; turn = 2;
      while (flag2 and turn==2) { };
      enterField; pickBerries;
      flag1 = false;
}
```

Model Peterson's algorithm for the two neighbors. Check that it does indeed avoid deadlock and satisfy the exclusion safety and berry-picking progress properties.

Hint: The following FSP can be used to model the turn indicator:

```
set CardActions = {set1,set2,[1],[2]}

CARDVAR = VAL[1],
VAL[i:Card] = ( set1 -> VAL[1]
              | set2 -> VAL[2]
              | [i] -> VAL[i]
              ).
```