Lab 6

Peterson's mutual exclusion revisited

The Peterson algorithm in its basic form ensures mutual exclusion but it does not also guarantee fairness. Threads can start re-competing for access to the critical section and overtake other threads advancing more slowly through the stages. A solution offering a certain degree of fairness can be obtained by slightly modifying the algorithm:

```
lock() {
        for (int L = 1; L < n; L++) {
                level[i] = L;
                victim[L] = i;
                while (( exists k != i with level[k] >= L ) &&
                        victim[L] == i ) {};
        }
        while ( exists k != i, level[k] != 0 &&
                                victim[level[k]] != k ) {};
}

unlock() {
        level[i] = 0;
}
```

The cause for being unfair in the original version of the algorithm lies in the fact that some threads might be slower than the others and spend more time until completing their stages. The effect of the "fix" is to slow down faster threads ensuring that slower threads advance.

Credits:

For detailed information and further enhancement of the Peterson algorithm please consult: *Fair and Efficient Mutual Exclusion Algorithms - K. Alagarsamy, K. Vidyasankar*.