Homework 5:  
Shared Variables

1. [Hoare triples] Derive the weakest precondition $P$ for the following program fragments:  

   i. $\{ P \} b := b \land c \{ b \Rightarrow c \}$

   ii. $\{ P \} y := y + 3z + 3x + 1; z := z + 2x + 1; x := x + 1 \{ y = x^3 \land z = x^2 \}$

   iii. $\{ P \} b := (b \equiv c); c := (b \equiv c); b := (b \equiv c) \{ b \equiv B \land c \equiv C \}$

2. [Shared variables] In his paper (Information Processing Letters, 12(3):115–116, 1981) Peterson makes the following remark about his algorithm:

   This algorithm does not appear ‘out of nowhere’, but is in fact easily derivable from simple forms. Consider the two primitive algorithms below. […] Both algorithms preserve mutual exclusion but both have deadlock. The first only when one process does not cyclically try and the second only when they both are trying. The waiting problems are disjoint and the correct algorithm is a simple combination of the two. The myth of difficulty of derivation is laid to rest.

   $q := 1; \quad q := 2; \quad [q = 2]; \quad [q = 1];$

   $CS1 \quad CS2$

   and

   $x1 := \text{true}; \quad x2 := \text{true}; \quad \neg x2; \quad \neg x1;$

   $CS1 \quad CS2$

   $x1 := \text{false}; \quad x2 := \text{false};$

   Make sure that you understand why each of these protocols on their own would not work. Now, another simple combination of these two programs is:

   $q := 1; \quad q := 2; \quad [\neg x2 \lor q = 2]; \quad [\neg x1 \lor q = 1];$

   $CS1 \quad CS2$

   $x1 := \text{false}; \quad x2 := \text{false};$

   Show that this solution of the mutual exclusion problem is not correct.

3. [The bakery problem] Consider a mutual exclusion algorithm based upon a scheduling method commonly used in bakeries and ice cream stores where each customer takes a number and waits for his/her turn.

   i. Assume an atomic “fetch and increment” statement:

   $$fai(s, p) \equiv \langle p := s; s := s + 1 \rangle \quad (p \text{ private, } s \text{ shared})$$

   Construct the algorithm using this instruction. Prove safety, progress and fairness.

   ii. Consider the following implementation of the bakery algorithm where the only atomic actions available are reads and writes. For notational convenience, we write
The program for the $i$-th process $P_i$ is given by:

```plaintext
\[
\begin{align*}
\text{[NCS]}; \\
c_i := \text{true}; \\
n_i := 1 + \max(n_0, n_1, \ldots, n_{m-1}); \\
c_i := \text{false}; \\
j := 0; \\
[j \neq m] \rightarrow \neg c_j; \\
\quad [n_j = 0 \lor (n_i, i) \sqsubseteq (n_j, j)]; \\
\quad j := j + 1 \\
\}; \\
\text{CS}_i; \\
n_i := 0
\end{align*}
\]
```

The shared boolean variables $c_0, c_1, \ldots, c_{m-1}$ are initially $\text{false}$, and the shared integer variables $n_0, n_1, \ldots, n_{m-1}$ are initially 0.

If the assignment involving the $\max$ function is implemented by the following program segment, the solution is not safe.

```plaintext
\[
\begin{align*}
\text{max} & := i; \\
k & := 0; \\
[j < m] & \rightarrow \\
\quad [n_{\text{max}} < n_k \rightarrow \text{max} := k] \\
\quad \lor [n_{\text{max}} \geq n_k \rightarrow \text{skip}] \\
\}; \\
k & := k + 1 \\
n_i := 1 + n_{\text{max}}
\end{align*}
\]
```

Construct a scenario that shows how the implementation fails.

iii. Fix the solution and briefly argue the correctness of your new version.

[Due to Leslie Lamport.]