GSI018 – SISTEMAS OPERACIONAIS

Operating Systems – William Stallings – 7th Edition
Chapter 05 – Mutual Exclusion and Synchronization

Nome Completo (Discente #1) – Nro. Matrícula – eMail Institucional
Nome Completo (Discente #2) – Nro. Matrícula – eMail Institucional

REVIEW QUESTIONS

5.3 What is the basic requirement for the execution of concurrent processes?

5.7 List the requirements for mutual exclusion.

5.8 What operations can be performed on a semaphore?

5.9 What is the difference between binary and general semaphores?

PROBLEMS

5.4 Consider the following program. a) Determine the proper lower bound and upper bound on the final value of the shared variable tally output by this concurrent program. Assume processes can execute at any relative speed and that a value can only be incremented after it has been loaded into a register by a separate machine instruction.
b. Suppose that an arbitrary number of these processes are permitted to execute in parallel under the assumptions of part (a). What effect will this modification have on the range of final values of tally ?

```
const int n = 50;
int tally;
void total( )  {
  int count;
  for( count = 1; count<= n; count++)  {
    tally++;
  }
}

void main( )  {
  tally = 0;
  parbegin (total (), total () );
  write (tally);
}
```

5.12 Consider the following definition of semaphores. Compare this set of definitions with that of Figure 5.3 . Note one difference: With the preceding definition, a semaphore can never take on a negative value. Is there any difference in the effect of the two sets of definitions when used in programs? That is, could you substitute one set for the other without altering the meaning of the program?

```
void semWait( s )  {
  if( s.count > 0 )  {
    s.count--;
  }
  else  {
    place this process in s.queue;
    block;
  }
}

void semSignal( s )  {
  if( there is at least one process blocked on semaphore s) {
    remove a process P from s.queue;
    place process P on ready list;
  }
  else
    s.count++;
}
```