

Homework 2 (due Wed. 6/9)

1. Consider the following serializer code:

```
serializer
{ queue a, b, c;
  crowd cr;
  int i = 0; /* initially zero */
  void foo() {
    enqueue(a) until i != 0;
    join_crowd(cr) {
      ... /* critical section */
    }
  }
  void bar() {
    enqueue(b) until a.empty() && i == 0;
    join_crowd(cr) {
      ... /* critical section */
    }
    i = 1;
  }
  void baz() {
    enqueue(c) until i == 1;
    join_crowd(cr) {
      ... /* critical section */
    }
    i = 0;
  }
} // end of serializer
```

Which of the following path expression(s) describe(s) only executions for which the above serializer does not deadlock?

- a) path { foo + bar + baz } + bar end b) path { bar ; foo ; baz } end c) path bar ; { foo } ; baz end
d) path { baz + foo } + bar end e) path { bar ; baz } ; foo end

2. This question has to be answered with respect to Solaris. Explain the performance impact of context switching from a thread T1 to a thread T2 in each of the following situations:

- a) T1 and T2 are both user level threads within the same process and are attached to the same LWP.
b) T1 and T2 are both user level threads within the same process but attached to different LWPs.
c) T1 and T2 are user level threads in different processes.
d) T1 and T2 are kernel threads unattached to any user level threads.

In each case, you should clearly identify the source of any performance hit that the thread switch would entail