The following assembly code used in lecture shows a potential race condition when two processes run **concurrently** to update a **shared** bank account balance (acct). Assuming an initial value of $1,000, the desired outcome is a final account balance of $480 (= $1000 - $500 - $20)

Process 1: acct = acct - 20	**Process 2: acct = acct - 500**
1: mov %r3,acct 2: sub %r3,20 3: mov acct,%r3	1: mov %r4,acct 2: sub %r4,500 3: mov acct,%r4

The following scenario of concurrent execution of both processes involves **two** context switches that end up with a final balance of $980.

Process 1: acct = acct - 20	**Process 2: acct = acct - 500**
1: mov %r3,acct 2: sub %r3,20 (INTR & Context Switch) 3: mov acct,%r3	 1: mov %r4,acct 2: sub %r4,500 3: mov acct,%r4 (INTR & Context Switch)

Show two additional scenarios of concurrent execution that yield incorrect balance in the shared account. Express your answer using a similar format shown above. Be clear to indicate where interrupts occur. Avoid using “interrupted AT line 3” (it is not clear where you mean **before** line 3 or **after** line 3)

(1) (3 pts) Involving **three** context switches (your answer should have 9 lines of “event”)

(2) (4 pts) Involving **four** context switches (you answer should have 10 lines of “event”)