Задача. *B. Tricky Mutex - 3 балла.*

Решение.
1) Mutual exclusion.
Рассмотрим потоки A и $B_1, B_2, \ldots, B_n$. Пусть, без ограничения общности, первым условие в while проверит поток A. Поскольку изначально $thread\_count == 0$, то поток A инкрементирует значение $thread\_count$ и выходит из цикла, захватив $mutex$. (При этом чтение и инкремент происходят атомарно) Далее все остальные потоки будут проверять и изменять переменную $thread\_count$, но она не станет равной нулю до того момента, пока A не выйдет из критической секции и не уменьшит значение $thread\_count$ на один. В этом случае $thread\_count$ может стать равной нулю, и тогда какой-то поток захватит освободившийся мьютекс. Но несколько потоков не смогут быть одновременно в критической секции, следовательно, такая реализация гарантирует взаимное исключение.

2) Deadlock freedom. Рассмотрим три потока A, B и C, которые пытаются захватить мьютекс. Пусть в начале $thread\_count = a > 0$ (например, некоторый поток D сейчас находится в критической секции). Рассмотрим следующую ситуацию:

$thread\_count$	A	B	C
a			
$a+1$	$fetch\_add(1)$		
$a+2$		$fetch\_add(1)$	
$a+1$	$fetch\_sub(1)$		
$a+2$			$fetch\_add(1)$
$a+1$		$fetch\_sub(1)$	
$a+2$	$fetch\_add(1)$		
$a+1$			$fetch\_sub(1)$
...	...	...	...

Так, в любой момент времени значение переменной $thread\_count$ больше $a > 0$, поэтому, даже когда поток D выйдет из критической секции, значение переменной $thread\_count$ будет не меньше числа a, которое больше 0. Следовательно, ни один из потоков не сможет захватить мьютекс. Такая реализация не гарантирует свободу от взаимной блокировки.