Question 2. Atomic or Not [6 marks]

The fetch-and-add hardware instruction *atomically* increments a value while returning the old value at a particular address. The C pseudocode for the fetch-and-add instruction looks like this:

```c
int fetch_and_add(int *ptr)
{
    int old = *ptr;
    *ptr = old + 1;
    return old;
}
```

The fetch-and-add instruction can be used to build a *ticket lock*, as shown below. When a thread wishes to acquire a lock, it first does an atomic fetch-and-add on the ticket value. The returned value is considered this thread’s “turn” (myturn). The shared value l->turn is then used to determine whether this thread can acquire the lock. When myturn == l->turn for a given thread, it is that thread’s turn to enter the critical section. The lock is released by incrementing l->turn so that the next waiting thread (if there is one) can now enter the critical section.

```c
struct lock {
    int ticket;
    int turn;
};

void acquire(struct lock *l)
{
    int myturn = fetch_and_add(&l->ticket);
    while (myturn != l->turn) // spin
    {
    }
}

void lock_init(struct lock *l)
{
    l->ticket = 0;
    l->turn = 0;
}

void release(struct lock *l)
{
    l->turn = l->turn + 1;
}
```

**Part (a) [4 marks]** Alice and Bob look at this code for a long time. Bob is convinced that the release code has a race (he suggests using fetch_and_add to increment l->turn to fix the race). Alice is convinced that the ticket lock code shown above is correct. Who is correct? Why?

Alice is correct. There is no race because l->turn is only updated by release(), and only the thread that calls acquire() calls release(). The acquire() function only reads the value of l->turn. It will either get the old or the new value of l->turn but this will not affect the correctness of the code.

**Part (b) [2 marks]** Assuming that Alice and Bob figure out how to implement the ticket lock correctly, would there be any benefit to using a ticket lock over a spin lock?

The ticket lock ensures fairness since each thread gets a ticket on arrival, unlike spinlocks. Also, ticket locks can be more efficient because they perform a regular read instruction instead of an atomic instruction while spinning in the acquire() code.