1 Thread-safe linked list

The most important change is the adding of a lock (`omp_lock_t`) to each node. This is later called "the lock". Each operation (even non-modifying ones as `count`) will grab and unset locks on the go. This is a design choice, to ensure we don't run into troubles wit non-atomic operations. The `count` (for example) could have problems reading a not-yet-ready pointer, and thus break correctness even if the program does not crash. Thus, each iterating function will lock a node, lock the next one, and only the unlock the former. This ensures we always hold a node, and thus will perform sooner or later.

`init_list` This should not be called concurrently, so no lock here. Only initiating the head's lock.

`append` We should lock the last element until the new node is initialized and currently last is updated.

`add_first` If two calls are made concurrently, one call should not be cancelled. Thus, we will not use `omp_set_lock` but `omp_test_lock` ; this allows us to read the head many times, until the first call to `add_first` has terminated and has updated the head. Then the second call will act on this new head.

`insert` We keep the node before the insertion locked. Then we create the new node, initialize it (with its next as the locked's next), modify our locked node (next is the created) and unlock. No need to keep more nodes locked.

`pop` Here we only hold the head until we can free it.

`remove_by_index` Here we hold not only the node but also its previous and next. So we can remove the node, and update the previous with the unchanged address of the next.

`remove_by_value` Exactly the same.

`count` We only lock an element, its next and unlock the previous to ensure every operation is finished before.

`search_by_value` Here again, we do the same as `count` as this is a non-modifying operation.

`print_list` Idem.

`delete_list` We must ensure no other call can be made to the list, thus we use `**head` and not `*head`. This allows us to set `*head` to `NULL` after deletion. Then we grab the next element, delete the current, shift current and always keep one node locked.