Concurrent Algorithms 2010: Exercise 10

1 Problem 1

Given a memory allocator M and a software transactional memory STM, extend STM with transactional memory allocation functions, preventing the memory leaks resulting from aborted transactions.

M is an object with two methods: `alloc(size)` and `free(ptr)`. `alloc(size)` returns a pointer to a block of memory of `size` bytes for the application to use. `free(ptr)` returns a previously allocated block of memory back to M, for use in subsequent *alloc* calls. *Memory leaks* occur when some memory block is allocated from M, but never freed.

STM is a software transactional memory that has standard functions `tx_start`, `tx_commit`, `tx_read`, `tx_write` and an internal `rollback` function that is invoked whenever a transaction needs to restart. (The function parameters are the same as in the lectures.)

Your goal is to implement two additional functions that the application can invoke: `tx_alloc(size)` and `tx_free(ptr)`. These functions are invoked whenever the application needs to allocate or free some memory transactionally (i.e. if the transaction aborts, the effects of `tx_alloc` and `tx_free` should be rolled back). It might be necessary to modify some of the STM functions as well.

2 Problem 2

Given a single global lock L, implement STM with the following API: `tx_start`, `tx_commit`, `tx_read` and `tx_write`. (The function parameters are the same as in the lectures.)

L is an object that has two functions `acquire` and `release`. `acquire` takes the ownership of L if it is not owned by any other process and returns. If it is owned by another process, it blocks the invoking process until the owner invokes `release`. `release` simply gives up the ownership of L. If some processes are blocked in the invocation of `acquire`, one of them will take the ownership and will proceed when the `release` is invoked.

Assumption: In both problems, processes cannot fail.