	points	Error description
NAME	10	wrong file name
COM	80	compilation error on a VM
FOL	5	folders inside zip file
gc1	-5	More reader than writers - gc should be barely active
gc2		
time	-5	timeout (the process run much time the the time limit)
max	-5	max is never reached
mem	-10	memory problem
gcp	-5	There is a print inside the critical section in the GC
nogc	-5	There is no GC in the code
fail	-20	Every run of the simulator failed
tclose	-10	Threads are not joined correctly or some threads are closed badly
mwfail	-10	Fail when running with more writers than readers
mrfail	-10	Fail when running with less writers than readers
timeout	-5	
DNS		did not submit
calls	-5	once or twice: didn't check return values of ALL system calls
calls2	-15	repeated: didn't check return values of ALL system calls
errno	-5	once or twice: On an error: print a descriptive error message (string + result of strerror(errno)) and exit.
errno2	-10	repeated: On an error: print a descriptive error message (string + result of strerror(errno)) and exit.
perror	-1	do not use perror
nam	-5	wrong name for one of two of the methods
nam2	-7	many method names are wrong
extra	-10	used extra functions of fields besides what was listed in the assignment
access	-10	only the 7 functions listed in the HW assignment are allowed handle the list fields in any way
alloc	-5	unnecesary allocation
stat	-5	List variables should be static inside a struct
recur	-5	didn't set mutexattr to PTHREAD_MUTEX_RECURSIVE
cond	-5	didn't create condition, or didn't do it correctly
init	-5	didn't initialize the list or did it incorrectly
clear	-5	didn't clear list and free all items before destroying
destroy	-5	didn't destroy lock or condition
destroy_pop	-5	do not use pop in destroy
implement	-5	error in implementing logic of a list function
implement2	-15	error in implementing logic of a at least 2 list functions
size	-5	should have a size variable which a function just returns without any lock or iteration and counting
lock	-5	extra lock or extra use of lock
locks	-10	several extra locks
no_lock	-5	missing a lock operation
push_head	-5	should malloc() the item before grabbing the lock. Then add to the list, release the lock. There should be a pthread_cond_signal() right before or after releasing the lock
push_head2	-10	made both possible mistakes in push_head error
pop_tail	-5	didn't follow this logic: grab the lock, remove item from list, release lock.

free	-5	didn't free the memory allocated for the item or freed before releasing the lock
empty	-10	When the list is empty, you should pthread_cond_wait() -- the condition should be in a while loop i.e., {{{ while (list_empty) pthread_cond_wait(&cond,&lock); }}}, it's a mistake to do it with an "if" (or no condition, or no wait()). If yourelease the lock and grab again whenever the list is empty that's also wrong
last_k	-10	shouldn't use pop_tail() directly. This is because the lock should be grabbed once! i.e., this function should grab the lock, remove k items (or less than k if the list has less than k), then release the lock. needs to also free the items it removes
timing	-5	push_head() should increase list size by 1 while grabbing the lock. pop_tail() should similarly decrease size by 1. move_last_k() as well, should decrease by k (or to 0 if there are no items). regarding pop/push - don't allocate or free the item within the lock
timing2	-5	repeated the error "timing"
main	-5	main should create all threads, then join them without deadlock. The correct way to do this is not to cancel() threads, but: stop the readers, join() the readers. Stop the GC, join() the GC. stop the writers, join() the writers (it's important to end with stopping the writers). Any other order, or cancel(), most likely causes deadlock (we will only assign the error if it does)
notify	-5	writer threads must notify the GC when the list is large enough, but only then
GC	-5	GC should grab the lock, and cond_wait() while list is less than max (can be <= or <, both ok). Again as pop_tail() the cond_wait() should be wrapped in a while loop, not if, etc. after the while, it should call remove_last_k() and then release the lock, and loop
finish	-10	main should make sure to call destroy on the list --- only after all threads finished.
deadlock	-5	allowed deadlock, usually by cancelling a thread which might hold the lock, or by joining the writers and then the readers
queue	-5	not OK if the main() adds items to the queue itself
wakeup	-5	readers or writers wakeup the gc more than necessary
wakeup_cond	-5	not OK if the main() calls the list's inner cond_variable
print	-5	missing a print or makes extra print. the GC should log not on every wakeup, but every time it works (i.e., actually removes items) - with a single print. the GC should print OUTSIDE the critical section, i.e., grab the lock, wait/cond until items are found, remove items, release the lock, print
print2	-10	makes more than one extra print