**Part B Documentation**

Student info:

Name: Minh Nguyen

NSID: thn649

Student #: 11286346

Partner:

Name: Joshua Rollo

NSID: apr213

Student #: 11214158

* Petgroom\_init():

We create 9 variables:

1. available\_stations: behaves like a semaphore. It checks for current number of available stations, decrementing by 1 if a pet is currently being groomed in newpet(), otherwise incrementing by 1 in petdone()
2. stationLock and condStation: the mutex and the condition variable for threads
3. DogsWaiting and CatsWaiting: the wait counter for dogs and cats, respectively. Increments by 1 while waiting for a station.
4. DogsSChed and CatsSched: counters for the number of dogs/cats which are scheduled to a station.
5. numCats and numDogs: counters for cats/dogs are being groomed in a station.
6. We also created a macro called SCHEDTHRESHOLD indicating the maximum number of dogs/cats can be scheduled to a station continuously before giving up the slots to the other. For example, if the threshold is 3 and the queue is “dog, dog, dog, cat, dog”. After 3 dogs, a station is reserved for a cat first, then give a slot for the following dog.

We returns -1 if pthread\_mutex\_init() fails to initialize the mutex and pthread\_cond\_init() fails to initialize the condition variable.

* Newpet(pet\_t pet):

We return -1 if pthread\_mutex\_lock/unlock fails to lock/unlock the mutex, or pthread\_cond\_wait fails.

While waiting for a station, we divide into 3 scenarios:

1. If pet argument is other: we just need to wait for available\_stations to be more than 0
2. If pet is cat: we need to check if the available\_stations is greater than 0, CatsSched is less than SCHEDTHRESHOLD, and numCats is equal to 0. If any condition is false, a cat should wait.
3. If pet is dog: we need to check if the available\_stations is greater than 0, DogsSched is less than SCHEDTHRESHOLD, and numDogs is equal to 0. If any condition is false, a dog should wait.

We then increase the number of cats being groomed in a station and the number of cats scheduled by 1. If we have reached the threshold for cats, for example, we let those cats finish being groomed then reserve all available slots for dogs, and vice versa.

* Petdone(pet\_t pet):

We return -1 if pthread\_mutex\_lock/unlock fails to lock/unlock the mutex, or pthread\_cond\_broadcast fails. For dogs, the if statement resets DogsSched counter back to 0 if there are no cats waiting for a station, and DogsSched is capped at the threshold. The same principles apply to cats. These if/else if statements actually cover another scenario: if there is a steady stream of dogs and no cats (or opposite), the sched counter resets in order to prevent from indefinite waiting time.

* Petgroom\_done():

Set available\_stations, numCats/Dogs, Dogs/CatsWaiting, Dogs/CatsSched to 0. We return -1 if pthread\_mutex\_destroy fails or pthread\_cond\_destroy fails.

* Grooming(void\* aPet):

Is a wrapper function to use in pthread\_create from the test program.