This project asked us to simulate lizards crossing a driveway while evading two hostile cats, creating a thread for each lizard and both cats. We then had to synchronize the lizards crossing the driveway in such a way that prevents them from being caught by cats. Some additional constraints included:

- We could not let too many lizards cross at once,
- Could not use busy loops to control the lizards, and we
- Had to allow the maximum number of lizards possible to simultaneously cross.

To accomplish this, we made a number of changes to the original code, the most important being the addition of semaphores. They allowed us to block lizards from running when it was unsafe for the lizards to cross, and unblock when the driveway was safe and clear, allowing the world to run at extended lengths without any lizards being caught (see Table 1). The comprehensive list of changes to the code includes:

1. Created a semaphore for lizard crossing
    a. Added logic to the lizard to check if the driveway is safe
    b. If a lizard is crossing then it blocks the other lizards
    c. Then unblocks, letting the other lizards know it is safe
    d. Once world as ended, deletes the semaphores
2. Code that created the cat threads, and joined them at the end
3. Code that created the lizard threads, and joined them at the end
4. All necessary instances of pthread_exit(NULL) to ensure the threads terminated nicely

In doing so, we face some challenges, which included:

1. The main issue stemmed from the use of semaphores, making sure that the critical section of the code - the number of lizards crossing - was protected so that no more than the allowed number of threads could access it.

2. The use of object-oriented programming when using semaphores is reliant on static methods because it more closely resembles the functional code found in C.

Table 1

WORLDEND (s)	**Maximum Number of Lizards Crossing**	**Lizards safe?**
30	4	Yes
70	4	Yes
110	4	Yes
150	4	Yes
180	4	Yes