Dan Aossey

Mutex: Mutext was a pretty easy thing to assure using a semaphore simply labeled mutex. I have a semaphore with a value of one that gets claimed with sem_wait before the critical section and is not released until the thread has finished everything within the passed in function with, sem_post.

Progress: Progress should be pretty easy to ensure because after the critical section is entered there is nothing between the sem_wait and sem_post lines that will cause the function to stop. I have an implementation of the safety algorithm that will decide if the current thread can enter the ride car without causing deadlock and thus between the two statements there should be no concern of stopping Progress. I will say I ran into a little difficulty where if you enter a weight limit that is so small no combination of attendees can enter the program will infinitely loop with no real progress being made and I could not find a good way to implement a solution to this problem.

Bounded Wait: I struggled with bounded wait because I understand it the goal is prevent starvation and in some cases it feels like starvation is inevitable. For a really obvious example, say you have a set up where a car can hold 4 people, the weight limit is 1000lbs and every person in the crowd weighs 250 except Steve who weighs 300. There is no combination of people where Steve gets to ride and the car is at full capacity and based on the prompt the car will not leave unless it is at full capacity. I tried implementing a couple ideas like a counter on each person to make sure that if someone got run a lot they might get turned away even if they make a safe state but none of these solutions worked in a case like the example above. Ultimately I kinda decided to go with the microsoft approach and ignore it. I see the possibility in a counter system like the one I tried a couple times but at least in this project I was unable to make it work and so far based on my testing I did not see much if any starvation in my threads.