**Studio 13**

1. **Names of the people:** Sayali Patil
2. **Why it is important that both of the processes that will be sharing the memory region see the same layout for this structure?**

Because since two processes are sharing the same memory region for read and write calls if the underlying data structure is not the same it can cause chaos while trying to coordinate the access of the two processes to the memory region.

1. **Explain whether you think element-wise assignment or using memcpy() is likely to be more efficient, and why?**

Memcpy() will be more efficient because it is much faster than element-wise assignment. Also, memcpy() is type-safe. Changing the data type to non-primitives also becomes easy with memcpy() as it calls for copy constructors.

1. **Explain briefly why it was necessary to remove both O\_CREAT and the call to ftruncate() from the follower program.**

As we know O\_CREAT is the option that we use to create a memory region if it doesn’t exist already. Since leader is already creating a region, and as the region is going to be used by the follower, we do not need to specify it in the follower code to avoid false creation of another region. Same with ftruncate() as we know it’s used to set the size of the memory region and as leader is already doing that, follower doesn’t need to.

1. **Explain briefly why you believe your concurrency protocol correctly avoids data races, deadlocks, and other hazards.**

Since we are using read, write, and delete guards to synchronize the actions between leader and follower programs using the same memory which make sure that leader does not take action before the follower action that needs to be completed before leader’s and vice versa helping avoid the data races, deadlocks, and other hazards.

1. **Output:**

Could not take output as linux lab isn’t working