The common characteristic between all projects was the difficulty of implementation. Understanding what was happening in this project was fairly simple, but implementing an environment that correctly emulates the virtual memory management system proved challenging. Over the course of working on this project, it started as a very simple signal reader program, to a fully correct virtual memory emulator of a First In First Out Replacement policy and an attempted Third Chance Replacement policy. We started this project by first implementing a working signal handler. This basic signal handler only allowed us to know when a fault occurred which gave us a basis for building the rest of the program. Next, we started setting up the the structs and variables that would be needed to implement fifo. The struct we used holds the information for each page in the m frames specified. This struct is then utilized in a double linked list to emulate the frames in virtual memory. Each node of the linked list operates as a frame. We push and pop these nodes based on the signals and eviction logic in the fifo signal handler. The fifo struct and the linked list functions that use the struct all correspond only to the fifo implementation of this program. The clock struct and the linked list functions that are used by the struct all correspond only to the Third Chance implementation of this program. We decided to keep all data and variables separate for the most part between the two different replacement policies. We did this minimize confusion in the implementation as well as to help us divide and work on the project without overwriting any working code. Fifo was the easier replacement policy to implement. This implementation was fairly simple for completing the eviction logic and protection policies, but we ran into minor problems with logging the correct data. Through debugging we were able to correctly output the evictions as well addresses and write-back. Once we were satisfied with the functionality of fifo we both moved onto the implementation of third chance replacement. This implementation proved much more difficult in terms of the eviction logic and the protections policies. Some of the hardest points to understand with this implementation was understanding how and when fault types 3 and 4 were occurring. We started the third chance policy by heavily using our implementation of fifo to build off of.

Both implementations utilized a double linked list. The linked lists allowed for us to add, remove, and manipulate the information stored within them as if they were the virtual memory frames. This also allowed for us to avoid allocating or moving any of the physical memory. The implementation of third chance specified that we use a circular linked list to complete it, so we set up functions that would go back to the beginning of the linked list once the list had been fully iterated through. Our fifo implementation adds pages to the beginning of the list and removes pages at the end of the list based on the signals received. Our third chance implementation utilizes the chances in the eviction logic to iterated through the list and evict the correct page in the frames.

For this project we wanted to work on each part as a unified yet separate team so that we could successfully complete each small portion of the project one by one building up to the implementation of the third chance replacement policy. We both independently worked on each portion of the code since both of us had roughly the same understanding of the project at the very beginning. Working independently allowed for each of us to try out our own basic implementations to start out with creating the signal handler and fifo replacement. We would use

the code of the person who had success first with the implementation we were currently working on. Doing this allowed for us to independently work and try to find success while keeping us both on the same page and maintaining the same code between the two of us.

Some additional information that we would like to include is that we were able to get fifo working for every input. Our third chance replacement policy is minorly working but does not correctly detect fault types of 3 and 4. We successfully implemented the circular linked list structure to keep track of the data for the third chance algorithm by iterating through if the tail of the list pointed to NULL we always jumped back to the head using a conditional and a global variable that always points to the head of the list. As we iterated through for the third chance algorithm we used mprotect if a second chance had occurred in order for the program to throw a fault so we can reach cases 3 and 4, but nothing occurred. Challenges we faced we initially starting as we had some issues conceptaulizing the beginning portion. Finding a way to use REG_ERR and understanding how X86 handles them was tedious. As well as using the signal handler as we had not used it before. Our third chance output is close apart from missing the fault types 3 and 4.