The project requires two C programs to communicate with each other using shared memory. The program receiver.c runs in an infinite background loop receiving alpha numeric values as input from the user, one line at a time. After reading one line from the standard input program then sends this information to the other program only if the line contains the secret code "C00L". The sharing of data between the two processes takes place via shared memory.

We write a function `create_shared_mem()` to create a shared memory and attach it to a pointer sh_ptr. We use a key while creating the shared memory (The key should be the same for both the programs) using the `shmget()` function. Once the shared memory is created, we get its id and save it in the variable `shared_memory_id.` Using the shared memory id, we attach the shared memory to a pointer `sh_ptr` (using the function `shmat()`), so as to be able to access the shared memory. Error handling codes in case of failures of any of the above operations have been added.

Once the shared memory has been created, we use the `check_and_push()`to run a background loop and keep accepting user input. It then checks the user input to see if the data contains the term C00L. If it contains the said term, it saves the data into the shared memory using the sh_ptr pointer.

The main function essentially calls the above functions.

The second program processor.c creates an output file secrets.out. It then increments the counter depending on the number of alpha numeric values in the array. The program first calls the function `get_shared_memory()`, which attaches the shared memory created by the receiver to `sh_ptr` (as we have used the same key, in this case 100). The program then reads the content of the shared memory and counts the number of alpha numeric values and records it in the variable counter. The received content and the counter value is then printed to the secrets.out file.

Steps to execute the programs on terminal:

1. gcc –c processor.c
2. gcc –o processor processor.o
3. gcc –c receiver.c
4. gcc –o receiver receiver.o
5. ./receiver
6. Open another terminal window at the same path
7. ./processor
8. Provide the user input
9. Check if the secrets.out file has been generated

NOTE: The count will include the 2 zeros in ‘C00L’

A sample input output snapshot is as follows: