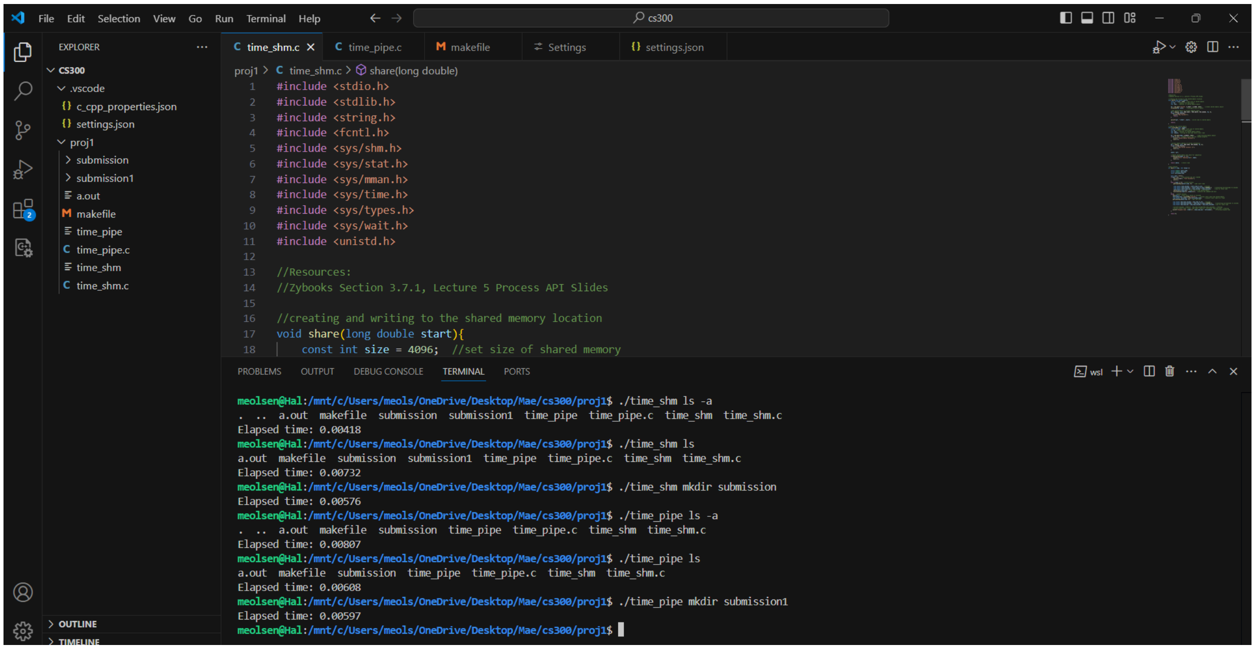
Programming Assignment 1

I. Screenshots of outputs



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <sys/shm.h>
#include <sys/stat.h>
#include <sys/mman.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

//Resources:
//Zybooks Section 3.7.1, Lecture 5 Process API Slides

//creating and writing to the shared memory location
void share(long double start){
    const int size = 4096;  //set size of shared memory
```

```
meolsen@Hal:/mnt/c/Users/meols/OneDrive/Desktop/Mae/cs300/proj1$ ./time_shm ls -a
.  ..  a.out  makefile  submission  submission1  time_pipe  time_pipe.c  time_shm  time_shm.c
Elapsed time: 0.00418
meolsen@Hal:/mnt/c/Users/meols/OneDrive/Desktop/Mae/cs300/proj1$ ./time_shm ls
a.out  makefile  submission  submission1  time_pipe  time_pipe.c  time_shm  time_shm.c
Elapsed time: 0.00732
meolsen@Hal:/mnt/c/Users/meols/OneDrive/Desktop/Mae/cs300/proj1$ ./time_shm mkdir submission
Elapsed time: 0.00576
meolsen@Hal:/mnt/c/Users/meols/OneDrive/Desktop/Mae/cs300/proj1$ ./time_pipe ls -a
.  ..  a.out  makefile  submission  time_pipe  time_pipe.c  time_shm  time_shm.c
Elapsed time: 0.00807
meolsen@Hal:/mnt/c/Users/meols/OneDrive/Desktop/Mae/cs300/proj1$ ./time_pipe ls
a.out  makefile  submission  time_pipe  time_pipe.c  time_shm  time_shm.c
Elapsed time: 0.00608
meolsen@Hal:/mnt/c/Users/meols/OneDrive/Desktop/Mae/cs300/proj1$ ./time_pipe mkdir submission1
Elapsed time: 0.00597
meolsen@Hal:/mnt/c/Users/meols/OneDrive/Desktop/Mae/cs300/proj1$ 
```

II. Based on my observations, message passing is faster than shared memory for each command I've tried because the time being shared is a relatively small amount of information to communicate.

A. Pros/Cons for shared memory

Pros:	Cons:
• Can be faster than message passing because no kernel intervention after object creation	• Controlling synchronization and security issues

B. Pros/Cons for message passing

Pros:	Cons:
• Good for exchanging smaller amounts of data because collisions aren't an issue	• Implemented using system calls, so kernel intervention requires more time than shared memory
• Easier to implement than shared memory	
• Easier to use in distributed systems	