There are 3 variants each for both programs P1 and P2.

String Generation: Random strings have been generated using rand() function. A string containing all the alphabets(upper and lowercase) has been created. rand() generates a random value whose remainder is taken with 51 and the alphabet corresponding to that index is added to the new string being created. In this code, strings of length 5 have been generated.

Unix Domain Sockets: Sockets are used to connect 2 nodes on a network so that they can communicate with each other. The server node (P2) listens on a particular port, while the client node (P1) reaches out to form a connection.
A Unix domain socket has been created using the **socket()** function, and is binded. It listens to requests and accepts request. In P1 socket command and connect has been used to connect by a socket pair. P1 generates the strings(and concatenates them with index at end) and uses **write()** function to send them as a single string(all 5 strings are concatenated together) to P2. P2 reads the strings using **read()** function and prints them to the console, and sends back the highest index receive to P1 through the same socket. P1 sends the next 5 strings starting from the successor of this index. **clock_gettime()** function has been used to get the total time taken to send all strings and receive acknowledgement.
To run this code, run './p1_socket & ./p2_socket' on bash after makefile finishes building

FIFOs: We have used 2 FIFOs in this question -p1_to_p2 and p2_to_p1. P1 writes the 5 strings to p1_to_p2 and P2 reads from it. P2 prints strings to the console and writes back the highest index received to p2_to_p1 and P1 reads from it and sends next 5 strings starting from that index's succesor. Reading and writing are done using **read()** and **write()** functions in C, FIFO is created using **mkfifo**. In this part, we have written the 5 strings one by one to the fifo instead of writing a concatenated string of all 5 strings. Time has been calculated using **clock_gettime()** function.
To run this code, run './p1_fifo & ./p2_fifo' on bash after makefile finishes building

Shared Memory: A shared memory object is created using **shm_open()** function and P1 writes 5 strings to it using **mmap()** function. P2 reads the strings using mmap() and prints them to the console. P2 writes the highest index received to the memory object using mmap and P1 reads it and sends next 5 consecutive strings. In this question structs have been used instead of directly using strings. The struct has 2 attributes -string and index. An array of structs is sent from P1 to P2 to send strings. Locking mechanism has also been implemented in this question using an extra struct in order to make sure that there is no race condition between P1 and P2. Time has been calculated using **clock_gettime().**
To run this code, run './p1_shared_mem & ./p2_shared_mem' on bash after makefile finishes building