In this part we are required to implement Inter-Process Communication.

For fifo:
In P1: In this part also first the batch of 50 random strings is generated with their indexes encapsulated in a struct.
Now a new fifo data structure is created in the tmp(cache memory of linux). The data is then first written into the fifo and after a batch has been sent. the client waits for the server's message. It then sends the next corresponding batch of random strings. In then end unlink command is used to free the fifo memory.

In P2: In this part as well the same thing is done. It first reads the values from fifo which the client wrote.
Then it calculates the largest ID string and writes the data back into fifo to send back to client.
The data is the read by the client and the loop goes on till 10 batches of random strings are not sent.

To run fifo: ./p2fifo & ./p1fifo

For sockets:
In P1: In this program in main function we define a variable of struct sockaddr_un by the name of clientSocket_addr.
Then we use the syscall socket which creates a new socket and has arguments AF_UNIX, SOCK_SEQPACKET, 0.
Then we set addr.sun_family to AF_UNIX. Then we copy the socket name to clientSocket_addr.sun_path which stores the pathname. Then we use the syscall connect to connect to the server.
Then I created 50 random strings inside the main function. I created a struct named pair which would store the string value as well as the corresponding ID. I created an array of size 5 and in every iteration
I am sending 5 random consecutive strings of length 6 from the 50 sized array.
Then we use the 'write' syscall to write to the client socket the array .
Then we wait to receive the index returned by P2 using 'read' syscall with args file descriptor and receiving array.
Then we close the socket.

In P2: In this program we first unlink any previous sockets if bound using 'unlink' syscall.
Then we use syscall socket to create a new socket. Then we set name.sun_family = AF_UNIX.
Then we copy the SOCKET_NAME in name.sun_path. Then we bind the socket using bind() syscall with a well known address so that the server can connect to it. Then we use the 'listen' syscall so that it will accept incoming connection requests. The listen command is used to accept data and make no. of queues till which it can accept data. Then we use 'accept' syscall to accept an incoming connection from the client.
Then we get the data using 'read' syscall and print it.
Then we choose the highest index and send it back to client using write() syscall.

Then we close the socket.

To run sockets: ./p2socket & ./p1socket

For shared memory:
In P1 : In this program we first created a shared variable using shmget() and shmat().
Then we generate 50 random strings and start sending them to P2 using the shared variable 5 at a time.
Then the code waits for the signal from P2 if they have recieved the string and then sends the next set of strings

In P2: In this program we use the same shared variable created in P1 using shmget() and shmat().
Then we receive strings from P2 and lock the shared variable until we receive the string and send to P1 the maximum ID received. SHared memory is the fastest IPC mechanism.

To run sharedmemory: ./p1shared & ./p2shared