Assignment 3

utils.h stores the common random string generator and struct StringEntry which is used to send the 5 arrays as one group. Random string generator is configured to generate random strings of alphabets only for simplicity. For all ascii characters, uncomment the line provided.

Sockets Implementation – p1_socket.c corresponds to P1 and p2_socket.c corresponds to P2. P1 imports utils.h and 50 random strings are generated and stored. After the socket is setup, P1 waits for P2 to request ID 0. Once P2 sends request for ID 0, P1 stores the 5 strings in the struct StringEntry and sends it P2. P2 receives, dereferences and prints the strings with their ids and sends back another request with highest id of the last group. This process is repeated till all the strings have been transmitted to P2. All necessary error handling is implemented.

How to run sockets implementation – Open 2 bash terminals. Use "make" command to compile the files in either of the terminals. Then run "./p1_socket" first in any one of the terminals. Then run "./p2_socket" in the second terminal. Order of running commands is essential here.

FIFO Implementation – p1_fifo.c corresponds to P1 and p2_fifo.c corresponds to P2. P1 imports utils.h and 50 random strings are generated and stored. After the pipe is setup, P1 waits for P2 to request ID 0. Once P2 sends request for ID 0, P1 stores the 5 strings in the struct StringEntry and sends it P2. P2 receives, dereferences and prints the strings with their ids and sends back another request with highest id of the last group. This process is repeated till all the strings have been transmitted to P2. All necessary error handling is implemented.

How to run fifo implementation – Open 2 bash terminals. Use "make" command to compile the files in either of the terminals. Then run "./p1_fifo" first in any one of the terminals. Then run "./p2_fifo" in the second terminal.

Message Queue Implementation – p1_queue.c corresponds to P1 and p2_queue.c corresponds to P2. P1 imports utils.h and 50 random strings are generated and stored. After the socket is setup, P1 waits for P2 to request ID 0. Once P2 sends request for ID 0, P1 stores the 5 strings in the struct StringEntry and sends it P2. P2 receives, dereferences and prints the strings with their ids and sends back another request with highest id of the last group. This process is repeated till all the strings have been transmitted to P2. All necessary error handling is implemented.

How to run queue implementation – Open 2 bash terminals. Use "make" command to compile the files in either of the terminals. Then run "./p1_queue" first in any one of the terminals. Then run "./p2_queue" in the second terminal. Order of running commands is essential here.

NOTE: If the queue implementation gives wrong or no output, please use "./clear_queue" and try running the implementation again