This project consists of extending the protocol developed in project 2 to enable it to deal with loss of messages. The E_S&G implements basically the protocol rdt3.0 presented in the textbook. The E_S&G consists of a client and a server. Communication is unidirectional, i.e., data flows from the client to the server. The server starts first and waits for messages. The client starts the communication. Messages have seq number 0 or 1. Before sending each message, a checksum is calculated and added to the E_S&G header. After sending each message, the client starts a timer (use alarm or sleep). When the timer goes off, the client tries to read a corresponding ACK message. If the corresponding ACK is not there, the message is sent again and the timer is started again. If the corresponding ACK is there, the client returns to the application which can now send one more message. This means that the E_S&G blocks on writes. The server, after receiving a message, checks its checksum. If the message is correct, the server sends an ACK0 or ACK1 message (according to the seq number) to the client and delivers the message to the application. If the message is not correct, the server repeats the last ACK message.

To verify your protocol, use the result of a random function to decide to send or skip a message, to decide to send or skip an ACK message, and to decide whether to send the right checksum or just zero. This will fake the error and loss effect.