#include <stdio.h>

#include <stdlib.h>

struct frame {

int info;

int seq;

};

int ak;

int t = 5, k;

int disconnect = 0;

struct frame p;

char turn = 's'; // Initialize first turn as sender

int errorframe = 1; // No Error

int errorack = 1; // No Error

void sender();

void receiver();

int main() {

p.info = 0; // Data part

p.seq = 0; // Sequence number

while (!disconnect) {

sender(); // Call sender

for (k = 1; k <= 10000000; k++);

// After a finite amount of time, call receiver

receiver();

}

return 0; // Add return type for main

}

void sender() {

static int flag = 0;

if (turn == 's') { // Sender's turn

if (errorack == 0) { // ACK didn't arrive

printf("SENDER: Sent packet with seq NO: %d\n", p.seq);

errorframe = rand() % 4; // Randomly pick error frame as 4

printf("%s\n", (errorframe == 0 ? "Error While sending Packet" : ""));

turn = 'r'; // Set next turn as Receiver for transmission

} else {

if (flag == 1) {

printf("SENDER: Received ACK for packet %d\n", ak);

}

if (p.seq == 5) { // If sequence number is 5, disconnect

disconnect = 1;

return;

}

p.info = p.info + 1;

p.seq = p.seq + 1;

printf("SENDER: Sent packet with seq NO: %d\n", p.seq);

errorframe = rand() % 4; // Random error in sending packet

printf("%s\n", (errorframe == 0 ? "Error While sending Packet" : ""));

turn = 'r'; // Set next turn as Receiver

flag = 1;

}

} else { // Time reducing process if turn is not sender

t--;

printf("SENDER: Time reducing\n");

if (t == 0) {

turn = 's'; // Return turn to sender

errorack = 0; // Simulate ACK error

t = 5; // Reset time

}

}

}

void receiver() {

static int frexp = 1;

if (turn == 'r') { // Receiver's turn

if (errorframe != 0) { // If no error occurred while sending packet

if (p.seq == frexp) { // If frame sequence number is equal to frexp

printf("RECEIVER: Received packet with seq %d\n", p.seq);

// Note sequence number of frame arrived to send acknowledgment

ak = p.seq;

// Increment the frame sequence number

frexp = frexp + 1;

// Set next turn as sender

turn = 's';

// Simulate error in sending ACK

errorack = rand() % 4;

printf("%s\n", (errorack == 0 ? "Error While sending ACK" : ""));

} else {

// Receiver received duplicated frame for lost frame after Resending

printf("RECEIVER: Duplicated packet with seq %d\n", frexp - 1);

// Note down acknowledgment number of frame

ak = frexp - 1;

// Next turn sender

turn = 's';

// Simulate error in sending ACK

errorack = rand() % 4;

printf("%s\n", (errorack == 0 ? "Error While sending ACK" : ""));

}

}

}

}