Project#1 (Due: December 8, 2020)

In this project, you are modifying the code given in the slides/textbook to implement the RDT 2.2 protocols using either TCP or UDP as the transport-layer protocol. You may use either C++ or Java to complete this project.

To simplify the project, we have the following assumptions:

1. Each packet will carry

Field	Size
Header - **Flag**	1 byte (0 0 0 0 0 0 0 1 - ACK, 0 0 0 0 0 0 1 0 - Data)
Header - **Sequence Number**	1 byte
Header - **Length**	1 byte
Header - **Checksum**	1 byte
Data	20 bytes

2. The initial sequence number is 0.
3. The sender will send out one packet at a time, and each packet carries the above info. Here the data field will store up to 20 characters (in its ASCII code.)
4. The checksum is calculated in the same way as TCP/UDP using all the information (including header and data).

For example, a data packet carries the following information.

Flag	Sequence Number	Length	Checksum	Data
0 0 0 0 0 0 1 0	1 1 0 0 0 0 1 0	0 0 0 0 0 0 0 1	**1 1 1 1 0 0 0 1**	0 1 0 0 1 0 0 0

then, the checksum is (**1 1 1 1 0 0 0 1**) based on the header information and the data.

5. When the **sender's** program starts, it provides two options: (1) normal outgoing packet transmission without error, and (2) random error bit. For the second option, the error bit is randomly selected and flipped after the checksum is calculated.

Note: The original packet should be stored in the buffer for retransmission.

6. Similarly, when the **receiver's** program starts, it also provides two options (1) normal outgoing packet without error, and (2) random error bit in the ACK. For the second option, the error bit is randomly selected and flipped after the checksum is calculated.
7. The sender may send up to 20 characters of data.
8. When the receiver accepts the packet (after the checksum is verified and has no error), it prints out the content of the packet and sends back an ACK without data (updates the length to 0.)
9. When the sender receives expected ACK, it prints out the message in the buffer and erases the content in the buffer.
10. The sender and receiver can keep on exchanging messages until the programs are terminated.
11. For simplicity, once the sender and receiver start, they keep their original options (with or without error.)

Rubric	Points
The client and server programs can be compiled and executed without errors	30
The client is fully functional in RDT 2.2	25
The server is fully functional in RDT 2.2	25
Write a report, which includes (1) how you design your programs, (2) challenges encountered during implementation, (3) how to compile and execute your code [OS , compiler, etc.], (4) other things you want to talk about this project.	20

References:

1. Beej's Guide to Network Programming (http://beej.us/guide/bgnet/)

2. Checksum algorithm (https://book.systemsapproach.org/direct/error.html?highlight=cksum#internet-checksum-algorithm)