Functions:

Download the **new** "`.h`" file from moodle, and put it in the same folder as the "`.ino`" file. You should import it using "`#import "EthernetLab5.h"`".
The functions:

- `setAddress(int number, int pair)` - Does all the initial Settings.
  - `number` - gets number 1 or 0, one Aruduino should be 1 and other 0.
  - `pair` - pair number from the moodle.
- `void setMode(int mod)` - set non-persistent or 1-persistent.
  - `mod = PRES` - set 1-persistent.
  - `mod = NON_PRES` - set non-persistent.
- `int checkLine(void)` - Returns 1 if the line is free, otherwise returns 0.

  This function should be called in the following events:

  a. Before transmitting a new message (i.e., check if the medium is taken).

     If the line is not taken, you may transmit the package.

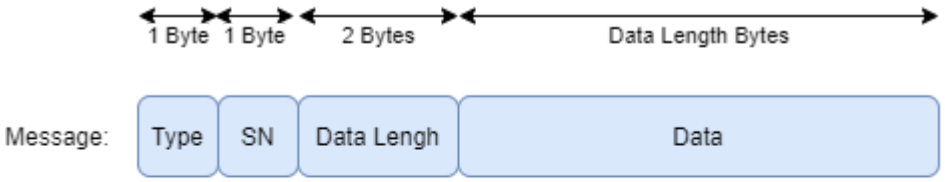
  b. While transmitting a frame to check if there is a collision.
- `void startPackage(void * payload, int payload_size)` - starts to send the frame to the server.
  - `payload` - a char array of data to send. Note that this array **has to casted** into a `void` array.
  - `payload_size` - the size of the data, E.g., char data[15] -> Maxdatasize = 15. We recommend using "`strlen`."
- `int endPackage(int option)` - called after propagation time (defined as 300 milliseconds).
  - If `option` = 0 - stops the frame mid transmission (in case of a collision).
  - If `option` = 1 - no collision detected and the transmission finished (i.e., successful transmission).
- `int readPackage(char* payload, int payload_size)` - reads frames, and also indicates if new data arrives by returning 1 (if there is new data) and 0 (if not).
  - `payload` - destination buffer of chars (the data is saved here).
  - `payload_size` - the maximal size of the buffer.

Implementation:

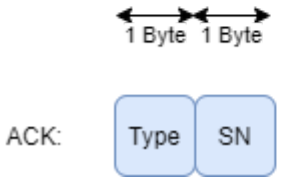
You need to implement CSMA/CD 1-persistent **and** CSMA/CD non-persistent with *Stop and Wait*. Note that you only need to implement the sender, and your code is half-complete from the previous lab.

Frame Structure:

The frame sent is an array of "`char`"s. The frame in this lab must be of the form:



Our server returns a frame of the form:



Field explanation:

1. Type - An indicator of whether the frame is a data frame or an ACK frame. If "Type" is 0, the frame is a data frame. If it's 1, it's an ACK. You should discard any other type. The server discards frames with non-zero types.
2. SN - Sequence Number.
3. Data Length - The length in Bytes of "Data." *Why, in your opinion, is this field necessary?*
4. Data - A field containing the information from the upper layers (your cool names).

CSMA/CD 1-persistent:

1. Use the function "`checkLine()`" to check the line; if the line is not taken, then transmit the message; if not, check the function continuously until it becomes idle and then transmit the message.
2. During the transmission itself, check the line if a collision occurred. If it did, stop and wait (pun intended) a random time (dubbed "Exponential Backoff") before retrying to transmit.

CSMA/CD non-persistent:

1. Use the function "`checkLine()`" to check the line; if the line is not taken, then transmit the message; if not - wait a random time before checking again.
2. During the transmission itself, check the line if a collision occurred. If it did, stop and wait for the Exponential Backoff before retrying to transmit.

Exponential Backoff:

Exponential Backoff is a mechanism to wait for a random number of time slots (TIME_SLOT length is a "`define`"). For the first time, wait $U_1 \sim Uniform\{0,\ 15\}$ time slot.

The next time waiting should be a random number of slots out of $\{0, 31\}$.

In the i^{th} round, wait for $U_i \sim Uniform\{0,\ 2^{i+3} - 1\}$ (discrete uniform random variable) amount of time slots. Finally, the amount of time to wait is $TIME\ SLOT\ \cdot U$.

In this lab, i caps at 7 (maximum of 1024 timeslots) and resets to 1 upon successful transmission (to a maximum of 15 timeslots).

This procedure is the same in both versions of CSMA/CD - it is very recommended to implement it once in a separate function.

General Tips:

1. **Read all general tips; they might help you with your code, questions, and defenses.**
2. Communication with the server only works if you are at the university while the controller is connected to the network.
3. Implementing Exponential Backoff is not as hard as you might think. Using the "`pow()`" function from "`math.h`" each collision can be tedious. It is more efficient to save "`i`" and update the upper limit in each collision.
4. "`random(min, max)`" returns a discrete-uniformly distributed number between "`min`" and "`max-1.`"
5. The time to wait (calculated by the Exponential Time function) should be "`unsigned long.`" Other types may overflow and cause bugs.
6. Some fields in the frame are 2 bytes or 4 bytes. You can use casting to put each field in the correct location. You can find examples in the previous Lab.
7. **Your code should be well documented!**