Project-1a Setup

1. Sign up for a GitHub account if you don’t have one yet.
2. Get the invitation to access the skeleton code. You can find it on Canvas.
3. Once you accept the invitation, you will see your own private repo here: 
   https://github.com/ucsd-cse123-wi21
4. Clone the repo and make sure the skeleton code works.
Overview

1. Implementing communication between two or more hosts
2. Hosts implemented as Threads
3. Network link is simulated
4. 2 types of hosts → Senders and Receivers
5. Sender hosts must transmit messages typed in at the command line to a corresponding receiver host.
6. Messages can be dropped
7. **GOAL: Implement reliable transmission!**
Code Structure
Code Structure

common.h
sender.c
Sender_t (common.h)

struct Sender_t {
    // DO NOT CHANGE:
    // 1) buffer_mutex
    // 2) buffer_cv
    // 3) input_cmdlist_head
    // 4) input_framelist_head
    // 5) send_id
    pthread_mutex_t buffer_mutex;
    pthread_cond_t buffer_cv;
    LLnode* input_cmdlist_head;
    LLnode* input_framelist_head;
    int send_id;
};

run_sender (sender.c)

1. Determine the next time the thread should wake up
2. Grab the mutex protecting the input_cmd/inframe queues
3. Dequeues messages from the input queue and adds them to the outgoing_frames list
4. Releases the lock
5. Sends out the messages
Code Structure
**Receiver_t** *(common.h)*

```c
struct Receiver_t {
   // DO NOT CHANGE:
   // 1) buffer_mutex
   // 2) buffer_cv
   // 3) input_framelist_head
   // 4) recv_id
   pthread_mutex_t buffer_mutex;
   pthread_cond_t buffer_cv;
   LLnode* input_framelist_head;
   int recv_id;
};
```

**run_receiver** *(receiver.c)*

1. Determine the next time the thread should wake up if there is nothing in the incoming queue(s)
2. Grab the mutex protecting the input_msg queue
3. Dequeues messages from the input_msg queue and prints them
4. Releases the lock
5. Sends out any outgoing messages
Code Structure
communicate.c

- Implements transporting of messages between the sender and receiver threads.
- Two main methods:
  - send_msg_to_receivers
  - Send_msg_to_senders
- Supports buffers of at most 64B
- Communication is **broadcast** based!
- Messages can be dropped!
- Messages can be corrupted! (not used in Project 1a)
Code Structure

- **input.c**
  - Handles messages inputted by the user
  - msg 0 1 hello world
- **main.c**
  - Handles command line options
  - Starts the sender threads, receiver threads, stdin thread
Tasks

1. Framing
2. Acknowledgements
3. Retransmission using timeouts
4. Stop-and-Wait scheme
Framing

- Divide messages into frames. Helps with error detection and retransmission
- Maximum frame size defined by MAX_FRAME_SIZE in common.h = 64B
- Create a header format

```c
#define MAX_FRAME_SIZE 64
#define FRAME_PAYLOAD_SIZE 64
struct Frame_t {
    char data[FRAME_PAYLOAD_SIZE];
};
```
Tasks

1. Framing

2. Acknowledgements
   a. Receiver should respond to the sender that it has received the corresponding message.

3. Retransmission using timeouts

4. Stop-and-Wait scheme
Tasks

1. Framing

2. Acknowledgements

3. **Retransmission using timeouts**
   
a. If a message is lost in transit, your senders should retransmit it after waiting more than 0.085 seconds but less than 0.1 seconds. We recommend **0.09 seconds**.

4. Stop-and-Wait scheme
Tasks

1. Framing
2. Acknowledgements
3. Retransmission using timeouts
4. **Stop-and-Wait scheme**
   a. Will be discussed in class. Or read the textbook!